AFRL-HE-WP-TR-2005-0094

AIR FORCE RESEARCH LABORATORY

Developing Standard Ontological Behavior Representations to Support Composability

William J. Gerber
Lee W. Lacy

Dynamics Research Corporation
3505 Lake Lynda Drive, Suite 100
Orlando, FL 32817

December 2004

Final Report for April 2003 to December 2004

Human Effectiveness Directorate
Warfighter Interface Division
Cognitive Systems Branch
2698 G Street
Wright-Patterson AFB OH 45433-7604

NOTICES

When US Government drawings, specifications or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Air Force Research Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Federal Government agencies and their contractors registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
8725 John J. Kingman Road, Suite 0944
Ft. Belvoir, VA 22060-6218

TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2005-0094

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

//SIGNED//

MARIS M. VIKMANIS
Chief, Warfighter Interface Division
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
December 2004	Final	April 2003 - December 2004

4. TITLE AND SUBTITLE
Developing Standard Ontological Behavior Representations to Support Composability

5a. CONTRACT NUMBER
F33615-03-C-6341

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
63832D

6. AUTHOR(S)
William J. Gerber, Lee W. Lacy

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
0476DM02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Dynamics Research Corporation
3505 Lake Lynda Drive, Suite 100
Orlando, FL 32817

8. PERFORMING ORGANIZATION REPORT NUMBER
E-9038U

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory, Human Effectiveness Directorate
Warfighter Interface Division
Air Force Materiel Command
Cognitive Systems Branch
Wright-Patterson AFB OH 45433-7604

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-HE-WP-TR-2005-0094

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Dynamics Research Corporation (DRC) supported the Defense Modeling and Simulation (DMSO) research initiative for Human Behavior Representation by investigating the use of the Web Ontology Language (OWL) for standardizing the representation of Computer Generated Force (CGF) behaviors within simulations to support composability for reuse. Currently, significant resources within the Department of Defense (DoD) are invested in developing CGF behaviors independently for various simulations. Many of the newer simulation systems are using hierarchies for representing simulations where simulation specific primitive behaviors are combined in a temporal framework to compose complex behaviors. The use of these composite behaviors, describing and relating the primitive behaviors in a formal ontology, results in sophisticated behavior representations and promotes reuse of behaviors, as they are themselves independent of the simulation implementation. This research project developed OWL ontologies for describing both primitive behavior metadata and composite behaviors and a prototype within the OneSAF Objective System (OOS) simulation demonstrating how they could be used for composing standardized behaviors in the future.

15. SUBJECT TERMS Ontology, Behavior Ontologies, Composable Behavior, Primitive Behavior, Composite Behavior, Web Ontology Language (OWL)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			1Lt R. Benjamin Hartlage
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	186	19b. TELEPHONE NUMBER *(include area code)* (937)255-9662

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

THIS PAGE LEFT INTENTIONALLY BLANK

ACKNOWLEDGMENTS

DMSO provided funding for this research with Air Force Research Laboratory (AFRL) providing technical direction as DMSO's agent. The following individuals provided invaluable support and advice: Dr. Sheila Benfield Banks, formerly from AFRL; Dr. Michael J. Young and 1Lt. R. Benjamin Hartlage from AFRL; Dr. Susan K. Numrich from DMSO; LTC John "Buck" Surdu, Bob Burch, and Dan Sagan (DRC) from the OneSAF Program Office; and Jim Blank, Dr. Andy Ceranowicz (Alion Science and Technology), and Bob Graebener (Institute for Defense Analysis) from the USJFCOM, J9 EED Modeling and Simulation Team.

ABSTRACT

Dynamics Research Corporation (DRC) supported the Defense Modeling and Simulation (DMSO) research initiative for Human Behavior Representation by investigating the use of the Web Ontology Language (OWL) for standardizing the representation of Computer Generated Force (CGF) behaviors within simulations to support composability for reuse. Currently, significant resources within the Department of Defense (DoD) are invested in developing CGF behaviors independently for various simulations. Many of the newer simulation systems are using hierarchies for representing simulations where simulation specific primitive behaviors are combined in a temporal framework to compose complex behaviors. The use of these composite behaviors, describing and relating the primitive behaviors in a formal ontology, results in sophisticated behavior representations and promotes reuse of behaviors, as they are themselves independent of the simulation implementation. This research project developed OWL ontologies for describing both primitive behavior metadata and composite behaviors and a prototype within the OneSAF Objective System (OOS) simulation demonstrating how they could be used for composing standardized behaviors in the future.

EXECUTIVE SUMMARY

DRC supported DMSO's Human Behavior Representation initiative by investigating the use of standard ontology behavior representations to support composability. DRC developed four generalized ontologies in the Web Ontology Language (OWL) for representing Computer Generated Force (CGF) behaviors. Those ontologies allowed for the composing of behavior instances in Resource Description Format (RDF) / Extensible Markup Language (XML) formats from primitive behaviors and other composite behaviors. (Primitive behaviors are simple behaviors hard coded in software but described with RDF/XML metadata while composite behaviors are more complex behaviors with temporal control of the execution of the included behaviors represented in RDF/XML data. Composite behaviors are also described with RDF/XML metadata.) The developed ontologies were:

- Behavior,
- Artifact,
- Variable, and
- ConceptDomainMetadata.

DRC developed a prototype behavior composer tool within the OOS simulation using the existing OOS graphical behavior composer tool. It was modified to allow transforming the XML behaviors created using the tool from the native OOS XML format into an RDF/XML format committed to the developed OWL behavior ontologies. The transformation was accomplished using Extensible Stylesheet Language Transformation (XSLT) files embedded within the prototype. The prototype tool also allowed the user to add metadata to the RDF/XML composed behavior instances that would represent the addition of Verification, Validation and Accreditation (VV&A) type data as well as data documenting the authoritative sources for the created behavior instances. The latter demonstrated the capturing of Knowledge Acquisition / Knowledge Engineering (KA/KE) information and tagging it to the developed behavior.

DRC added an additional capability, a simple filtering method that would allow a behavior developer to reduce the search space for the particular primitive or composite behavior needed during the composition. Currently, an OOS behavior developer would have to either "brute-force" search through all the available behaviors (166 in the OOS Build 15), checking each for applicability, or have the name of the specific behaviors to use already known/memorized. With the filtering based on instance data conforming to the ontologies, the captured RDF/XML instance data allowed the user to reduce the search space to only the behaviors that might be applicable, a much more manageable set.

The XSLT files for the ontologies were then used with the 166 OOS Build 15 behaviors formatted in XML to create instance files of those behaviors in RDF/XML formats compatible with the ontologies. They were then validated to confirm that they conformed to the ontologies.

For the JSAF option that was exercised, a couple of the behaviors from the vmove_task.fsm file were hand-coded, using the logic represented by the finite state machine (FSM) C-code, into RDF/XML instance files committed to the behavior ontologies. Those instance files were then validated to confirm that they conformed to the ontologies and that the JSAF behaviors could be suitably represented by them. To complete the loop, DRC developed another XSLT file that transformed the JSAF RDF/XML instance files back into the same "C-language" type of constructs in which they had originally appeared in the vmove_task.fsm file.

In a final ad hoc check to illustrate the capability of moving composed behaviors between disparate simulations, the JSAF XSLT was modified and used on some of the OOS behaviors that had been transformed into the RDF/XML formats. The output looked similar to what one would expect in the JSAF finite state machine format.

TABLE OF CONTENTS

LIST OF ATTACHMENTS

1 INTRODUCTION

Currently, significant resources within the Department of Defense (DoD) are invested in developing Computer Generated Force (CGF) behaviors for various simulations. Unfortunately, that entails high costs for developing new behavior representations for each new simulation or behavior and repeatedly verifying, validating, and accrediting those behaviors in their different forms. The Defense Modeling and Simulation Office (DMSO), through the Air Force Research Laboratory (AFRL) as technical agent, sponsored Dynamics Research Corporation (DRC) to (1) investigate the use of ontologies for describing both primitive behavior metadata and composite behaviors and then (2) demonstrate how they could be used for composing standardized behaviors in the future. The overall research goal was to enable the reduction of time and resources required in the future for both the development of CGF behaviors and the Verification, Validation and Accreditation (VV&A) of those CGF behaviors through enhancing the capability to reuse CGF behaviors between and within simulations. This report describes the research activities performed by DRC for developing standard ontological behavior representations to support composability.

1.1 Background

Simulations represent behaviors for CGF systems in various ways. Among the variations, hierarchies are used for some of the more recent CGF systems developed by the U.S. Army, such as the Close Combat Tactical Trainer (CCTT), WARSIM, and the OneSAF Objective System (OOS).

1.1.1 Behavior Hierarchies

Hierarchies are used to represent behaviors with a building block approach. The lowest level of behaviors, the atomic behavior, often provides the physics-based actions, and is normally implemented in simulation-specific software code. Those atomic behaviors are then composed into complex behaviors by combining them in a temporal framework. There, they can act in concert, such as in parallel, in sequence, or in various configurations including branches or loops where the path to the next behavior is

decision-based and considers what is occurring or has already transpired within the simulation. (Lacy and Henninger, 2003)

The higher level of behaviors, the composite behaviors for example, are more abstract than the primitive behaviors and can be described using data rather than software code (Lacy and Henninger, 2003). Thus, they can be simulation independent, allowing modification of composite behaviors without needing to recompile the underlying simulation's software code. The composability information that is part of the description of a composite behavior includes references to the component primitive behaviors, which are already a part of the simulation's compiled software. These primitive behaviors enable the composite behavior's actions to occur within the simulation. Those references to the primitive behaviors, such as name and version, are subsets of the primitive behavior's own descriptive metadata and allow selection of the correct primitive behavior that is a part of the simulation's software.

Thus, standardizing the primitive behavior metadata is important in supporting reuse of composite behaviors across simulations since primitive behaviors usually are implemented differently within each simulation. With a sufficient description of each primitive behavior that is to be used by a composite behavior, reuse of the composite behavior with a different simulation could occur, provided each primitive behavior on the other simulation had a description that matched closely enough.

1.1.2 US Army CGF Systems

CCTT, WARSIM, and OOS are three major U.S. Army CGF systems that represent recent implementations of CGF systems to support constructive and virtual simulations used for training and analysis (Lacy and Henninger, 2003). Their approaches to developing and representing behaviors use hierarchies.

The CCTT system is a virtual simulation system developed for training armor units at the company and platoon level (Law and Moerk, 2003). It uses Combat Instruction Sets (CISs), stored in a database by unit types as validated descriptions of U.S. Army doctrine, for the software engineers developing the simulation (McEnany and

Marshall, 1994; Ourston, Blanchard, Chandler, Loh, and Marshall, 1995). They use the CISs to develop Finite State Machines (FSMs) encoded in the Ada software language. The FSMs implement the primitive behaviors described in the CISs (Lacy and Henninger, 2003).

WARSIM is a constructive simulation being developed to train commanders of higher-level echelons (Lacy and Henninger, 2003). It employs a very extensive Knowledge Acquisition/ Knowledge Engineering (KA/KE) effort to describe the U.S. Army domain to software engineers. Using the KA/KE products (Lacy, 1997; Sagan, 2000), they develop Behavior Description Frames (BDFs), composite behaviors written in XML files, and fundamental behaviors, primitive behaviors written in C++ software code. Additionally, WARSIM uses a type of fundamental behavior called a WARSIM predicate function to provide Boolean values for decision making and to trigger activities (Karr and Holbrook, 1999; Foster, 1997).

OOS is a constructive level CGF system being developed to provide platform-level support to simulations (Henderson and Grainger, 2002). Like WARSIM, OOS has a very extensive KA/KE effort to develop KA/KE products for the developers (Anderson and Henninger, 2002). OOS is using the OOS Behavior Composer tool to develop composite behaviors, represented in XML and made up of primitive behaviors and other composite behaviors (DaCosta, 2002). The primitive behaviors are being written in Java and also include the concept of primitive predicate functions.

These three simulation programs all utilized KA/KE processes to develop validated descriptions of doctrine prior to behavior development. However, their use of different implementation approaches hinders reuse of their developed behaviors in other simulation systems. A standardized means of capturing both the doctrinal domain and the resultant behaviors would greatly aid the reusability of the results of the KA/KE efforts and, consequently, the ease of behavior composability and the reduction of VV&A effort.

1.1.3 Related Research

A variety of past research efforts have explored the development of ontologies for primitive behavior metadata. Some have focused on taxonomies, which are often represented by simple subsumption hierarchies, i.e., the "is-a" relationship. This is a simpler representation construct than ontologies, which can relate classes with multiple types of relationships and has been the focus of other efforts. (Lacy and Henninger, 2003) A summary of some of those projects is included next.

An object-oriented representation for describing tasks was developed as part of the Joint Simulation System (JSIMS) Joint Conceptual Models of the Mission Space (JCMMS) style guide work. In it, tasks were related to actors performing the tasks and other information, including inputs, outputs, and constraints. (JCMMS UML Style Guide, 1997)

As part of DMSO's Common Semantics and Syntax (CSS) effort, the Knowledge Integration Resource Center was developed (KIRC, 2004). Among its tools is the Functional Description of Mission Space (FDMS) Syntactic Category Search Tool, which allows searches for the definitions of nouns (entities) and verbs (behaviors). The verbs, which can be considered as the actions part of a behavior, are related in a tree-structured taxonomy. This verb taxonomy could be used to consistently name primitive behaviors. Similarly, Michael Fineberg proposed a general theory of behavior linking behaviors to their antecedents and consequences (Fineberg, 1995, 1998). His taxonomy focused on differentiating verbs into taxons and could be useful in identifying closely related primitive behaviors.

The Universal Joint Task List (UJTL) was developed to provide a standard language for describing high-level military operations and provides a means for documenting military task ontologies with Army doctrine (CJCSM 3500.04C, 2002). That would allow traceability from doctrine through KA/KE Artifacts to the derived primitive behaviors and their implementations, a definite support for verification and validation efforts.

In June 2001, DMSO hosted a workshop as part of their Common Human Behavior Representation and Interchange System (CHRIS) initiative to explore how to provide an authoritative, fully structured description and interchange mechanisms of human behavior that would lead to complete and unambiguous representations for military modeling and simulation (Lacy and Henninger, 2003). The "After-Action Report" recommendations (Bjorkman, Tyler, and Barry, 2001) included:

- Begin with an effort on Human Behavior Representation (HBR) Ontology development,
- Use XML and the DARPA Agent Markup Language (DAML) for behavior description,
- Track the technology evolution of semantic web research,
- Investigate the benefits of reusing models with greater abstraction, and smaller scope (vs. very large, very detailed models),
- Specify a common grammar (initial taxonomy and lexicon) for expressing behaviors, and
- Develop a standardized interchange format for behavioral knowledge.

"Enhanced reuse and standardized definitions and behaviors" were anticipated benefits identified by the group.

Another initiative, the Extensible Modeling and Simulation Framework (XMSF) which focused on defining a composable set of standards for web-based modeling and simulation, has identified the potential of the DARPA Agent Markup Language (DAML) technologies to consistently classify modeling and simulation data and services via precise vocabularies (Lacy and Henninger, 2003).

In summary, although a variety of methods can be used to represent and relate primitive behavior structures and content, the trend in research is towards neutral interchange mechanisms for sharing and reusing behaviors (Lacy and Henninger, 2003).

1.2 Semantic Web Representations

1.2.1 Ontologies

New technologies for representing domains through formalized ontologies are evolving. Sir Tim Berners-Lee, in 1999, set forth his Semantic Web vision (Berners-Lee, 1999) involving the addition of explicit semantics to information represented on the web. This evolution allows software to extract information directly rather than relying on humans to read the hyper-text markup language (HTML) and then transcribe it into a useable form for another application. DARPA has been investigating the use of a formal XML language, called DAML, for describing the semantic web concept through ontologies. That work, ongoing for several years, was combined with the Ontology Inference Language (OIL) as DAML+OIL and is being transitioned by the World Wide Web Consortium (W3C) into the Web Ontology Language, also called OWL. (Lacy and Henninger, 2003) On February 10, 2004, the W3C status of OWL was changed to a Recommendation, "understood by industry and the Web community at large as a Web standard." (W3C Press Release, 2004) That status is the same W3C status as for XML, now widely used and accepted.

1.2.2 Potential to Support Reuse

As mentioned earlier, the primitive behaviors are generally simulation specific and software coded. However, the metadata that describes them can be represented using data that is simulation independent. Using a formal ontology to describe and represent these primitive behaviors would promote their reuse. (Lacy and Henninger, 2003)

Using semantic web technologies to standardize the description of primitive behaviors, developers could then describe their primitive behaviors so that implementation independent composite behaviors could be developed and reused. Similarly, an ontology could be developed to also represent composite behaviors. (Lacy and Henninger, 2003)

The successful implementation of semantic web ontologies could result in the anticipated benefits of reduced CGF behavior development costs and a quicker

development time for validated composite behaviors. These results would primarily be realized by promoting the reuse of composite behaviors that standardize their references to primitive behaviors. Describing the primitive behavior metadata using semantic web ontologies would standardize those references.

1.3 DRC Scope

DRC performed research to begin to develop the next generation of behavioral representation methods in military simulation systems that support the concept of composable behaviors. DRC developed sample ontologies to represent human behaviors and programming metaphors and developed prototype software to illustrate the manipulation of ontology instances and the advantages of OWL over XML. DRC evolved behavior representation methods from XML to RDF/XML using OWL ontologies and used OneSAF behaviors to help scope the effort. For this effort, seven tasks were included. Task 6 was administrative in nature, designed to document the research activities in monthly reports and in this final report.

1.3.1 Task 1 - Requirements Analysis

This task was to determine the ontologies and tool functionality required to support a behavior representation tool. The Requirements Document deliverable was previously submitted and is again included with the electronic submission of this final report for completeness.

1.3.2 Task 2 - Design Ontologies

This task was to design military operation ontologies to support the specification of military primitive and composite behavior representations. The final Ontology Design Document is included as Attachment 1 – Ontology Design Document and separately as a Microsoft Visio document with the electronic submission of this final report.

1.3.3 Task 3 - Encode Ontologies

This task was to encode the behavioral ontologies using the World Wide Web Consortium (W3C) Ontology Web Language (OWL), and annotate sample behavioral

data according to the encoded ontologies. The resulting OWL ontology files are included as Attachment 2 – Behavior Ontology, Attachment 3 - Variable Ontology, Attachment 4 – Artifact Ontology, and Attachment 5 - ConceptDomainMetadata Ontology. Sample instances representing military primitive and composite behaviors are included electronically in the separate CDROM addressing the next task for Rapid Prototype Development.

1.3.4 Task 4 - Rapid Prototype Development

This task was to develop a prototype demonstration tool in Java. The tool was used to demonstrate composing behavior representations based on ontological primitives. The software for the prototype demonstration tool is included electronically on a separate CDROM, which is a separate deliverable.

1.3.5 Task 5 - Apply Prototype to Simulation

This task was to apply the behavior development approach and prototype tool to an opposing forces simulation system to be defined and coordinated through the JFCOM J9 office. The behaviors transformed into the RDF/XML format that conformed to the developed OWL behavior ontologies are included electronically in the separate CDROM addressing the previous task for Rapid Prototype Development.

1.3.6 Optional Task – JSAF Work

This task was to apply and instantiate the concept of composable behaviors to a specific Joint Semi-Automated Forces (JSAF) behavior or group of JSAF behaviors to be determined by the program office. The selected behaviors were to be transformed into RDF/XML files that conformed to the developed ontologies and then translated back into JSAF behavior content in coordination with the USJFCOM, J9 EED Modeling & Simulation team. The XSLT file that was developed for the transformation is included in Attachment 23 - JSAF_main.xslt. The activities associated with this task are described in the results section of this document.

2 RESULTS

The following sub-sections summarize the results from this research, including:

- Requirements analysis,
- Ontology design,
- Ontology encoding,
- Prototype development and demonstration,
- Application to OOS,
- Application to JSAF, and
- Program management activities.

2.1 Requirements Analysis

The requirements analysis was documented in the Software Requirements Specification (SRS), which was submitted as a deliverable on September 2, 2003. Primary areas that were identified for development were ontologies for describing primitive behavior metadata, ontologies for describing composable behaviors and demonstration software that would show how composite behaviors could be developed in the future. The requirements were described in two major sub-sections, one oriented to the ontology requirements and the other to the prototype for the next generation behavior composer. Both sections relied heavily on the OneSAF Objective System (OOS) behavior composer developments as a starting point.

Figure 1 shows a notional depiction for the prototype behavior composer tool interface (based on OOS) and provides insight into the temporal relationships that were to be encoded in the ontologies. The large rectangles represent temporal containers for components, which can include primitive behaviors (ovals), composite behaviors (rectangles), control branches (diamonds), as well as other temporal containers. The containers control their included components to execute either in parallel or in sequence. In Figure 1, the outermost rectangle is a parallel container that indicates its included primitive behavior would execute simultaneously with its included sequence container. That included sequence container, when it executes, would execute its own sequence container followed by a control branch that would either loop back to the sequence

container or end its execution. The innermost sequence container would execute its first primitive behavior on the left, test the condition for the control branch to execute one of the two primitive behaviors shown next, and then finally execute the fourth primitive behavior. When the last component within a container has finished, the container is considered to have finished. Once a container included within a higher-level sequence container has finished, the next component can start to execute.

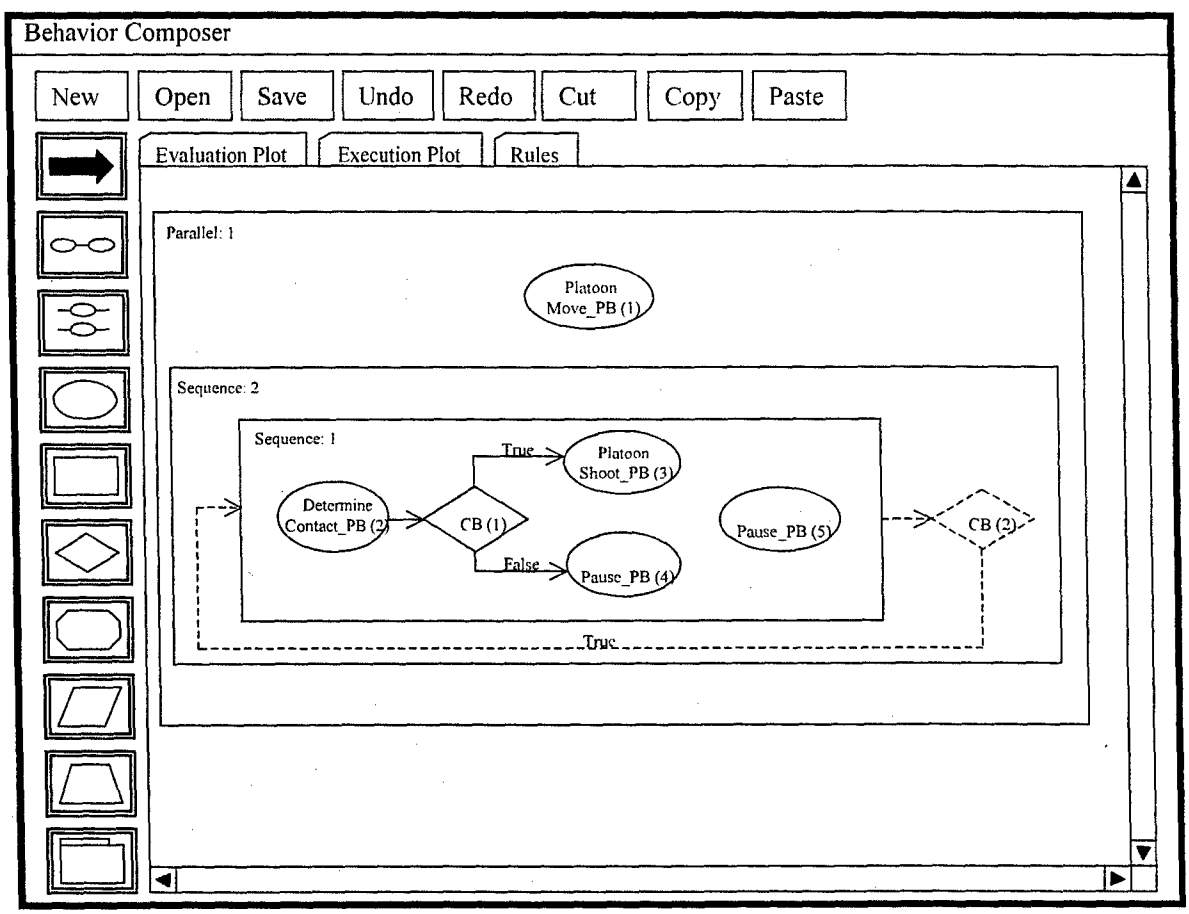


Figure 1. Prototype Behavior Composer Tool Interface

The Figure 2 context diagram depicts an overview of the objectives of this research and directly led to the SRS requirements for the Prototype Behavior Composer / Editor Tool. The tool should be able to read existing Primitive Behavior Metadata and Composite Behaviors and then allow the tool user to edit them or create new composite

behaviors. The user should then be able to store those read/edited/created primitive behavior metadata and composite behaviors in implementation independent RDF/XML files that are committed to the ontologies being developed under this research in OWL. That would support reusability by allowing other commercial/government off-the-shelf OWL-compliant tools to be able to access and use those previously developed behaviors. The OWL-based domain ontology, also being developed under this research, would allow the user of the prototype composer/editor tool to filter the behaviors being retrieved from the RDF/XML databases using search criteria based on the intended new behavior to reduce the time needed for composing that new behavior.

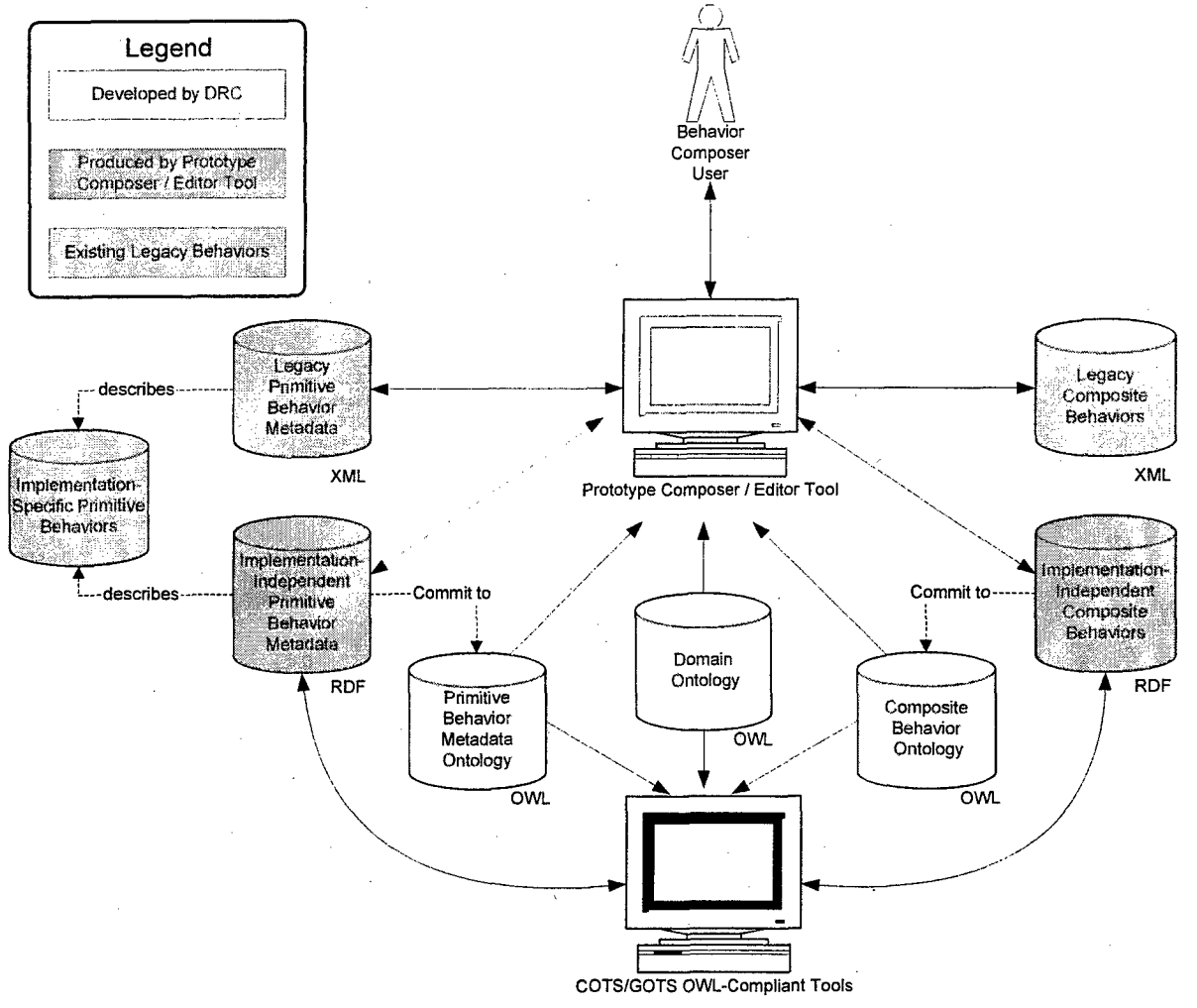


Figure 2. Context Diagram for Requirements

2.2 Ontology Design

The initial ontology design was documented in Microsoft Visio using Unified Modeling Language (UML) diagrams and submitted as a deliverable, Interim Report – Ontology Design Diagram, on October 31, 2003 with 17 ontologies. The current, final version of the ontology design is reproduced in Attachment 1. The ontologies have been consolidated and refined such that there are now only four– the basic Behavior ontology plus three supporting ontologies. They will each be described briefly in turn. Within each ontology, the classes in the UML diagram are depicted with data type properties shown within the class (labeled rectangle) and object type properties shown by labeled arrows from the subject class to the object class. Figure 3 is an example showing a portion of the ontology design diagram for the GeneralMetadata class which is defined in the Artifact ontology. Subclass relationships are depicted by a line from the subclass to a triangle attached to the superclass. Note that the Behavior class, defined in the Behavior ontology, is a subclass of the Artifact class and would thus inherit its data type and object type properties.

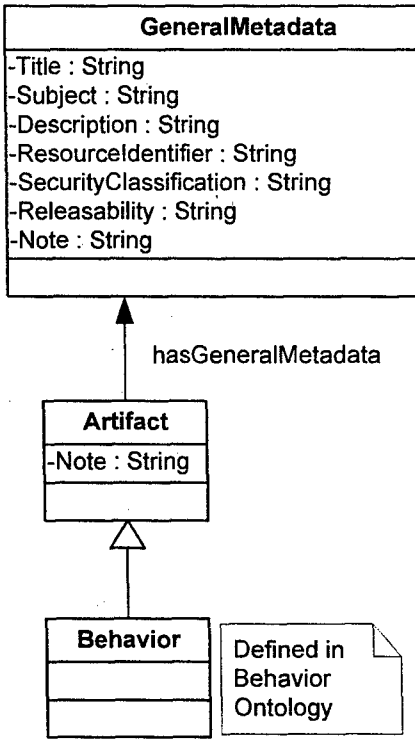


Figure 3. GeneralMetadata Class Example UML Design Diagram

The four ontologies being used are the basic Behavior ontology plus three supporting ontologies: Artifact, ConceptDomainMetadata, and Variable. As can be seen in Attachment 1, the Behavior ontology inherits classes and relationships from all three supporting ontologies and the ConceptDomain ontology also inherits from the Variable ontology.

2.2.1 Behavior Ontology

Note that the Behavior class is a subclass of both the Artifact and ContainerComponent classes, inheriting from both. It has subclasses of both PrimitiveBehavior and CompositeBehavior, each of which likewise inherits the properties from the classes of Behavior, Artifact and ContainerComponent. The ContainerComponent has subclasses of Container, ConditionalBranch, and Behavior which includes, through its subclasses, both PrimitiveBehavior and CompositeBehavior.

The ContainerComponent class represents the classes which can be placed within the Container temporal control element. Through the Container's object type properties of hasPrimitiveBehavior, hasCompositeBehavior, hasConditionalBranch, and hasContainer, the components of a Container can be defined. To capture the data flow, or order with which the components within a Sequence Container must execute, the object type properties of hasTrueBranch and hasFalseBranch for the ConditionalBranch class and hasNextComponent for the other container components are used. The Container also uses hasFirstComponent to indicate which component is executed first. These properties, plus the object type property hasSourceComponent inherited from the class ContainerComponent, provide for a two-way linked-list type structure, although only the forward linked structure is being used. To indicate which Container controls the execution control flow at the highest level, the CompositeBehavior class uses the object type property of hasTopLevelContainer.

To differentiate between the sequence and parallel containers, the Container has a Boolean valued data type property of ParallelContainerType. For a parallel container, all its components execute together, so they wouldn't use the object type properties indicating order.

The CompositeBehavior class allows for the case of having to direct behaviors for other entities which may be subordinates or having to send information to them through the Boolean data type properties of OrderSender and DirectiveSender. The object type property of hasSubordinatesRetrievalPredicate allows for retrieving the identities of subordinates to which commands or information must be sent.

The CompositeBehavior class allows for interrupts through the class of RulesPredicatesRepresentation, which has a subclass of LocalReactionRule. The hasConsequent property allows description of which Behavior to transition to if the current behavior has to be stopped. The Boolean data type property of GlobalReactionRule allows for limiting interrupts to specific portions of the behavior. Those portions can be defined using the subclass of LocalReactionRule and its object type properties of effectiveBefore, effectiveDuring, or effectiveAfter which can be associated with specific container components.

Both the classes of RulesPredicatesRepresentation and ConditionalBranch use the ConditionalExpression class to return a Boolean value for the decision of whether to interrupt or of which container component is next to execute, respectively. Note that the ConditionalExpression class has three subclasses, each with their own operator types, and can be used to represent complex expressions which must be evaluated to return a value. For simple cases, the value may be determined by calling a PrimitiveBehavior with hasConditionalPredicate, by calling for the value of an Input with hasInput, by evaluating the ConditionalExpressionStatement data type property expressed as a string, or by using the DefaultValue data type property, a Boolean expression.

2.2.2 Artifact Ontology

The Artifact class, as a superclass to the Behavior class, allows the Behavior class to inherit the metadata represented in the KeyWord, GeneralMetadata, and Version classes which are connected by object type properties. The versioning and VV&A information is captured through the Version class which has object type properties to the Authority and VVARecord classes.

The Artifact class also has the KAKEArtifact subclass which allows for capturing metadata about the authoritative KA/KE documents used in development of the Behavior through the derivedFrom object type property linking the Behavior to the KAKEArtifact.

2.2.3 ConceptDomainMetadata Ontology

The ConceptDomainMetadata class uses hasAcceptableOrganizationalLevel, has AcceptableSide, and hasAcceptableEntityType object type properties to capture what echelon level(s), side(s), and type(s) of entity can use the behavior. The Actor class, a subclass of the Entity class, describes the entity that will perform the behavior. The hasMechanism and hasOtherEntity object type properties allow specification of what equipment and other actor entities may be required for the behavior to be valid. The interplay of object type properties connecting the Entity, Unit, Side, and Task classes allows for a rich specification of interconnected relationships.

The ConditionsDomain, ConditionsLevel, and Descriptor classes allow for capturing conditions reflected in the Universal Joint Task List (UJTL) that may have a bearing on the validity for use of the behavior. For example, some conditions captured in the Descriptor class can be required to be either present or absent for the behavior to be valid while others can be described as having no consequence for the validity.

The ConceptDomainMetadata class also allows for specification of hasInput or hasOutput object type properties. However, none of the behaviors to date have used these properties and they may be extraneous.

2.2.4 Variable Ontology

The Variable superclass has Input, Output, and LocalVariable subclasses which differ in data type properties of Source, Destination, and Location, respectively. The object type properties of usesValueFrom and usedForValueOf, both of which predicates have the Variable class as both subject and object, allow for specific instances of behaviors to specify which variable values are to be used in the call to the externally defined behavior. For example, two instances of an externally defined behavior A used in a composite behavior B as behavior A1 and A2, may each need to have the variable V,

that is a defined input of behavior A, use different values, V1 and V2, for behavior instances A1 and A2 respectively. Also, for example, within a composite behavior, they allow the Output value of one behavior to be specified as the Input value for another behavior or as the value for a LocalVariable. Other combinations of value substitutions could be used.

2.3 Ontology Encoding

Initially, the ontologies from the Ontology Design Diagram were planned to be encoded using Network Inference's Construct visual modeling environment, an overlay on the Microsoft Visio application. That tool had the capability to automatically convert the UML model diagrams into OWL ontology code without further manipulation. Early results with the demonstration tool were encouraging for modeling the ontologies visually and having the tool convert the model into code. In fact, during an introductory trial period, the entire ontology design diagram was input in two days of work. However, DRC was unable to negotiate a license in sufficient time to use the tool. Instead, a database tool developed inhouse by DRC for creating DAML ontologies was modified to create OWL ontologies and renamed OWL DB Tool.mdb. Then the data necessary to encode the ontologies using OWL was input into the tool. The use of that tool eliminated many errors which would have been encountered if the entire set of ontologies would have had to be encoded manually. Once the data was entered, the DRC tool was able to print out the OWL ontology files.

To validate the ontology files, the online vOWLidator application from the BBN link, http://owl.bbn.com/validator/, was used. Each of the ontologies correctly validated using that link.

To encode the required sample behavioral data, the example behavior shown in Figure 1 was hand coded as an RDF/XML instance file committed to the set of developed behavior ontologies. It was then successfully validated using the World Wide Web Consortium (W3C) RDF validator.

The deliverable, Interim Report – OWL Ontology Files, was submitted on November 26, 2003. The four current behavior ontology files (Behavior, Variable, Artifact, and ConceptDomainMetadata) are at Attachments 2, 3, 4, and 5, respectively. Each of those ontologies was also successfully validated against the vOWLidator application from the BBN link.

2.4 Prototype Development and Demonstration

The prototype demonstration tool was initially planned to be a stand-alone tool using the OOS Behavior Composer (BC) software. The OOS Build 15 software was obtained in early January 2004 and loaded on an available laptop computer. Unfortunately, the numerous linkages between the OOS BC and the rest of the OOS Framework prevented decoupling the OOS BC software from the rest of the OOS Framework to allow using it in a stand-alone manner. However, once it was determined that menu items could be added to the OOS BC interface and interact with the OOS BC, the decision was made to build the prototype demonstration tool within the existing OOS BC rather than as a stand-alone tool.

The Saxon parser was installed within the OOS system to enable Extensible Stylesheet Language Transformation (XSLT) parsing of the OOS XML formatted behaviors and specialized Graphical User Interfaces (GUIs) were developed to make the demonstration prototype work as desired. Due to time constraints and a lack of detailed knowledge about what specific requirements had to be met by XML formatted behaviors for the OOS BC to work with them, DRC was unable to determine what would have to be done to the RDF/XML formatted behavior instances to transform them back into OOS XML formatted behavior instances that would work in the OOS BC. By carefully mapping the OOS XML elements into the equivalent RDF/XML elements, however, DRC was able to determine that the team could develop XSLT files for a small subset of the selected OOS behaviors chosen for the demonstration and illustrate the key concepts.

Considering the limitations of time and resources, the following concept was devised for how the prototype would work and what it would do. The OOS BC would be used in its normal capacity to compose behaviors. However, when a composed behavior

was selected to be saved as an RDF/XML instance, the behavior would be saved first as is normally done in the OOS Framework as an XML file. The user would then be presented a Java GUI to enter whatever additional metadata should be added to the behavior file that was to be saved. A small subset of the possible metadata was used in developing the GUI and what would be allowed for entry was restricted to that subset. When the user was finished entering the selected metadata, the metadata would be written out to a temporary XML file. The XSLT, under control of the Saxon parser, would then parse that temporary metadata file and the composed XML behavior file to create a new RDF/XML file that was the equivalent of the composed XML file, but with the metadata added. The system would then keep track of which XML behavior file matched the newly created and saved RDF/XML behavior file. Thus, whenever the user selected a newly created RDF/XML behavior file to modify for a new composition, the corresponding OOS XML behavior file would be presented in the OOS BC. To the user, that would be the equivalent of working with the new RDF/XML file.

To illustrate what more could be done through adding metadata to a behavior, two additional Java GUIs were created (one for primitive behaviors and one for composite behaviors) that would allow the user to filter for behaviors that matched the user's search criteria entries. Normally, whenever a user of the OOS BC wanted to instantiate a specific behavior for the generic one placed on the OOS BC canvas, the user was presented with every existing behavior of either the primitive or composite behavior type that the user was trying to use and then had to manually search for the specific behavior to use. There was little to guide the user to make the correct choice. With the filtering capability, the number of behaviors presented could be significantly reduced and inappropriate ones that did not match what the user was looking for would be screened out.

Those concepts were successfully implemented and demonstrations were held at the DRC Orlando Field Office for DMSO, AFRL and OneSAF on April 5, 2004 and at the OneSAF Program Office in Orlando on April 8, 2004. More detailed information on the demonstrations and the developed software are on a separately submitted CDROM.

The software contained on that CDROM could be used with the OOS Build 15 software to duplicate the prototype tool with the latest version of the ontologies and XSLTs.

2.5 Application to OOS

After the prototype tool demonstrations, the mappings between OOS behavior elements and the RDF/XML instance elements were expanded to cover all the types of constructs observed in the 166 Build 15 OOS behaviors. Constructs for Order Sender composite behaviors, Conditional Branch elements, Post Conditional Loop elements, and various Conditional Expression possibilities were addressed. The XSLTs were restructured and updated to handle all 166 OOS Build 15 behaviors and added back into the prototype tool. Those updated XSLTs for transforming the OOS XML behavior files into the simulation independent RDF/XML behavior files are in Attachments 6 through 18 as well as on the separately submitted CDROM. The 166 transformed RDF/XML formatted behaviors are on the separate CDROM under the two folders, \compositions\behavior\hbr\rdf\composite and \compositions\behavior\hbr\rdf\primitive. All of those RDF/XML formatted behaviors have been validated against the latest, current, updated behavior ontologies.

2.6 Application to JSAF

The behavior representations in the Joint Semi-Automated Force (JSAF) simulation reside in the finite state machine (FSM) files, those ending in .fsm, in sections of code that start with a "back tick", such as in:

```
` START
`{ },

` END
`{ },

` DRIVING
` tick { }
` params { },
```

where the description of the behavior is entered between the curly braces, { }. During compilation, those sections of code are compiled into C-code task files that have the same names but end in the .c suffix instead of the .fsm suffix.

2.6.1 Compilation From .fsm to .c Files

To observe what typically changes during compilation, the vmove_task.fsm file from the libvmove library was examined as well as its compiled version, vmove_task.c. For the vmove_task.fsm, the states were: START, CHASING, DRIVING, ARRIVING, ARRIVED, STOPPED, STUCK, DISABLED, and END. Except for START and END states, each had both a `tick { } and a `params { } event, where the tick behavior is called at each simulation time tick and the params behavior is called when parameters change. The two event types are asynchronous with respect to when they may be called. In the compiled file, the START and END fsm code is inserted directly into the body of the code for vmove_start and vmove_end functions with only one type of change, which occurred for the START behavior and not the END behavior. That type of change had to do with a state change and applied to all the other states as well and will be discussed after a description of their changes. For the other states, the code that followed each of the `tick events were placed into a vmove_tick function as code for a switch's case statement with the case name the same as the state name. In each case, the code was inserted directly with the only change being related to a directed state change in the code. The same compilion format followed for the code for each of the `param events with each of them being inserted into a switch's case statement in the same manner as the `tick code but into the vmove_params function instead.

The state changes in the vmove_task.fsm code are represented as a caret, ^, followed by the next-state name in capital letters, followed by a semicolon, and then followed by a comment, all on the same line. For example, the change to the CHASING state in the `DRIVING `params code is shown as

```
^CHASING; make our own route
```

which is compiled in the vmove_params C-code under the "case DRIVING" statement as

{ state->state = CHASING; break; }

without the comment but including assignment of the new state and a break; statement following it. In some cases where the state change was actually back to the same state, the state assignment did not show up but the break; statement was still shown at the end.

2.6.2 Representation of Behaviors in RDF/XML

Understanding what the ^STATE; representation meant allowed the behavior representation to be understood since the remainder of the code was actually normal C-code that did not change during the compilation. From these observations, it was determined that only the behavior representations in the vmove_task.fsm needed to be addressed for developing a representation in the RDF/XML format that would be equivalent. However, to represent some of the C-code, a convention was adopted that C-code statements would be represented as PrimitiveBehaviors. For example, the C-code statement of A = B; could be represented as a PrimitiveBehavior notionally named AssignValue with A being an Output variable and B being an Input variable. Similarly, the C-code Bitwise-And operator (&) used in a conditional statement A & B could be represented by a PrimitiveBehavior notionally named BitwiseAnd with both A and B as Input variables and the object of its returnsValue object type property being the Output variable with a value that would be the result of the Bitwise-And (&) operation.

Once it was determined how the behavior representations could be made, the specific behaviors had to be selected. The libvmove library was selected to find a subset of behaviors to be explored due to the researcher being somewhat familiar with the libvmove library from ModSAF. Due to the limited time available, two relatively simple behavior representations with a fair amount of internal variety were selected for the task of being hand-coded into the RDF/XML format. The START state and the params event of the DRIVING state were selected. The vmove_task.fsm behavior representations for these two behaviors are shown in Attachments 19 and 20

2.6.3 Naming Convention for Behaviors in RDF/XML

Next, a naming convention was adopted for introducing some regularity into the RDF/XML representation for naming the various behaviors. The first portion will always be "JSAF_". If the function is global in scope, i.e., not specifically limited to behaviors or functions in a specific behavior related library, it will have the next portion of the name with capitals and lower case and will end with PB for Primitive Behavior, e.g., "JSAF_AssignState_PB" or "JSAF_AssignValue_PB". If the function is limited in scope to a specific library or associated specifically to the library, it will have the second portion with that library name in lower case, e.g., "JSAF_vmove_" for libvmove, followed by the descriptive name with underline separations and will end with "PB" for PrimitiveBehavior. For example, to represent the libvmove functions of vmove_update_private or vmove_no_input_route, the names used would be "JSAF_vmove_update_private_PB" or "JSAF_vmove_no_input_route_PB".

For the finite-state-machine state behavior names, which are associated with specific libraries, the state name portion will be in all capital letters followed, when applicable, by lower case "_tick_" or "_params_", and end with "CB" for Composite Behavior, e.g., "JSAF_vmove_START_CB" or "JSAF_vmove_DRIVE_params_CB". Each specific composite and primitive behavior is placed in its own individual file using the name of the behavior followed by a ".rdf" suffix, e.g., the "JSAF_vmove_DRIVE_params_CB" behavior is located in file "JSAF_vmove_DRIVE_params_CB.rdf".

Since the behavior ontologies were not specifically designed for writing code, normally used functions within code writing, such as "=" for assigning a value, are handled within the behavior ontologies as primitive behaviors, e.g., "JSAF_AssignValue_PB", with appropriate inputs and outputs. (Those primitive behavior substitutes can later be decoded by the XSLT to reproduce the function in its normally used C-code form.)

2.6.4 Hand-Coding of RDF/XML Behaviors

Initially, the plan had been to use the prototype behavior composer tool to code the JSAF behaviors and then automatically save them as RDF/XML files. Theoretically, it would have been possible since the algorithms for the JSAF behaviors could be reproduced in the tool. Preliminary attempts revealed that the memory limitations of the laptop on which it was installed and the limited display space would make the operation more cumbersome than hand-coding the RDF/XML directly, particularly since the primitive behaviors and composite behaviors referred to would have had to be hand-coded in the prototype anyway to be able to use them. Additionally, and most significantly, at the time that the tool was needed, the XSLTs on the tool had not been completely updated to the latest ontology and would have produced uncertain results.

Hand-coding of the first RDF/XML instance file and the supporting instance files to which it referred was then undertaken. The first behavior had six behaviors associated with it. To code the behavior file JSAF_vmove_DRIVING_params_CB.rdf and then validate it, the following behavior files had to also be hand-coded.

(1) JSAF_vmove_CHASING_CB.rdf,
(2) JSAF_vmove_END_CB.rdf,
(3) JSAF_vmove_no_input_route_PB.rdf,
(4) JSAF_vmove_update_private_PB.rdf,
(5) JSAF_AssignState_PB.rdf, and
(6) JSAF_BitwiseAnd_PB.rdf.

All seven behavior instance files were validated against each other and the behavior ontologies using the BBN validator web link at http://owl.bbn.com/validator. Note that the composite behaviors, (1) and (2) above, were not internally complete, but had the behavior inputs and outputs instantiated to allow the composite behavior being developed, JSAF_vmove_DRIVING_params_CB, to be completely validated.

The JSAF_vmove_DRIVING_params_CB.rdf behavior instance was later redone to place the ConditionalBranch for individual "if" statements that did not have an "else" component in a Container class to allow simpler XSLT code for transforming the RDF/XML code back into the native fsm format. The new instance file was again

validated using the same validator link. The JSAF_vmove_DRIVING_params_CB.rdf file is at Attachment 21.

The second selected RDF/XML behavior file, JSAF_vmove_START_CB.rdf, was also hand-coded and the four supporting instance files to which it referred were also hand-coded. They were:

(1) JSAF_AssignState_PB.rdf,
(2) JSAF_vmove_update_private_PB.rdf,
(3) JSAF_AssignFunctionValue_PB.rdf, and
(4) JSAF_AssignValue_PB.rdf.

All five behavior instance files were validated against each other and the behavior ontologies using the BBN validator web link at http://owl.bbn.com/validator. Note that the first two primitive behaviors, (1) and (2) above, were previously hand-coded for the validation of the JSAF_vmove_DRIVING_params_CB.rdf behavior instance. The JSAF_vmove_START_CB.rdf file is at Attachment 22.

2.6.5 XSLT for RDF/XML Format to JSAF Format

To begin the XSLT effort for transforming the RDF/XML formatted behaviors back into the original JSAF formats, an algorithm for mapping the RDF/XML class structures and relationships into the fsm code was developed. From that, the generalized XSLT to perform the transformation needed to produce the fsm code was then coded. The XSLT is at Attachment 23 as JSAF_main.xslt.

The attempt to transform the JSAF_vmove_DRIVING_params_CB.rdf behavior instance into the fsm format was successful. The output can be seen at Attachment 24. The only structural difference between the original code and what was produced using the XSLT on the RDF/XML instance file for the behavior was that the XSLT introduced code blocks, i.e., sections bracketed by curly braces, "{" and "}", to contain the "if" and any "then" portion of code that would follow the "if" portion. Functionally, the two were equivalent.

Additionally, the JSAF_vmove_START_CB.rdf file was successfully transformed into the fsm format. The output can be seen at Attachment 25. Again, the only structural

difference between the original code and what was produced using the XSLT was the set of curly braces for the code block associated with the "if" portion of the code as previously described. Functionally, the two were equivalent.

As an aside, a quick attempt was made to modify the JSAF_main.xslt to make it more generic so that it could attempt to transform the OOS RDF/XML behaviors into JSAF formats. The attempt was not completely successful due to time constraints, although many of the OOS RDF/XML behaviors would partially transform with errors, mostly with respect to incorrect placement of the entire set of input variable data in the function arguments. Those errors were similar to errors encountered along the way in the development of the XSLT to transform the JSAF RDF/XML behaviors and could probably have been corrected if there had been time to work the issue.

2.7 Program Management Activities

A variety of activities were performed under this effort that supported the management of the project and satisfaction of specific contract requirements. The following sections describe the meetings supported, reports and deliverables provided, and the project website.

2.7.1 Meetings

A variety of meetings were supported and attended in support of these efforts. These meetings included:

- AFRL/DMSO PRDA Kick-off Meeting for Human Performance (Human Behavior Representation) at the IDA Building, Alexandria, VA, July 15-16, 2003.
- USJFCOM/J9 EED M&S Team Meeting with DRC for JSAF Option Task Coordination at USJFCOM, Suffolk, VA, October 7, 2003.
- AFRL/DMSO Project Review at DMSO, Alexandria, VA, January 29, 2004.
- Prototype Behavior Composer Tool Demonstration for AFRL/DMSO at the DRC Field Office, Orlando, FL, April 5, 2004.
- Prototype Behavior Composer Tool Demonstration for the OOS Program at the OneSAF Program Office, Orlando, FL, April 8, 2004.
- Demonstration of Prototype Behavior Composer Tool at BRIMS 2004 Conference, Arlington,VA, May 17-20, 2004.

- DMSO Technical Review at DMSO, Alexandria, VA, July 19, 2004.
- Paper Presentation on project and results at I/ITSEC 2004, Orlando, FL, December 6-9, 2004.

2.7.2 Reports / Deliverables

DRC provided various reports as deliverables. These included combined monthly technical progress and cost reports, presentation materials for presentations that were made, a Software Requirements Specification, an Interim Report – Ontology Design Diagram, an Interim Report – OWL Ontology Files, and this initial Final Report. Additionally, a Prototype Behavior Composer Tool Demonstration was provided as a deliverable. Documentation of the presentation, along with the software for that demonstration prototype, is being submitted separately as a portion of this initial Final Report. The 166 OOS Build 15 XML behaviors, all of which were transformed into RDF/XML formats that were validated to conform to the developed behavior ontologies, are also being submitted separately as a portion of the software deliverable.

2.7.3 Project Website

DRC maintained a project website throughout the effort (see Figure 4). The website is hosted at http://Orlando.drc.com/hbr and includes sections on:

- Program Objectives, to describe the lower-level project objectives,
- Latest Updates, for what is published on the site,
- Knowledge Base, for published information from the Kick-off Meeting,
- Restricted Area, for other program related presentations not viewable without a logon name and password, and
- Contact Us, a link to send email to the DRC research team.

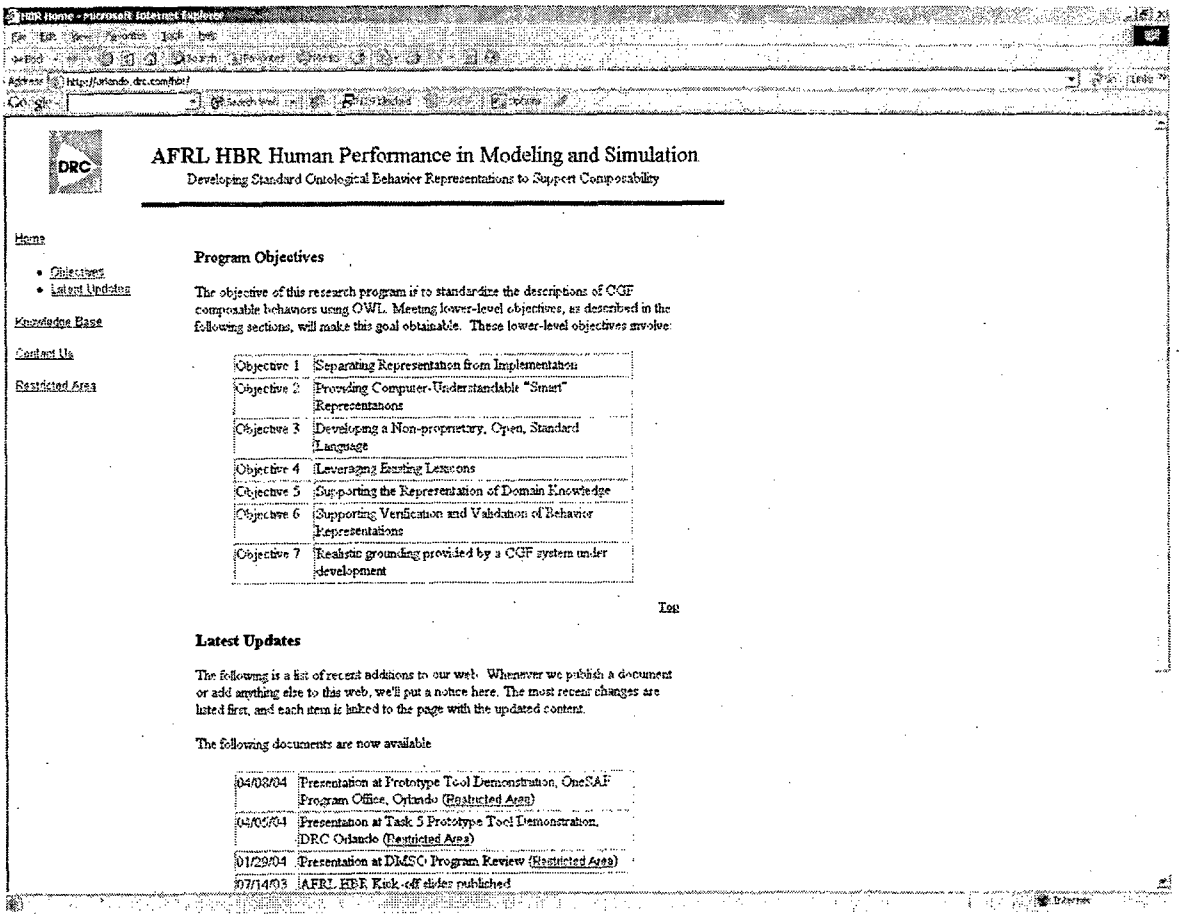


Figure 4. DRC Project Website

3 FUTURE DIRECTIONS

One future direction for research might involve trying to evolve and standardize what the behavior ontologies should include by broadening the base of simulations that are considered. Additionally, an expanded visibility of the behavior ontologies to more simulation behavior developers would undoubtedly improve them and provide greater assurance that what should be represented is actually included. A working group under the auspices of SISO or some other appropriate forum would be appropriate to further the standardization of behavior representations.

Another development that would be useful is a stand-alone graphical behavior composer tool that could be used by JSAF, OOS, WARSIM, or any other simulation to graphically compose behaviors and produce RDF/XML behavior instance files that would conform to a standardized set of behavior ontologies. The composed behaviors could then be transformed, using a suitable XSLT, into a JSAF C-coded fsm representation, an OOS XML representation, or any other specific simulation's type of representation. The primitives would have to be unique to the specific simulation using the behavior, but the metadata could describe what the primitive would have to do in sufficient detail so that corresponding primitives could be developed in the appropriate software code for any simulation that would need to use the composed behavior. Such a tool would also be useful for needed, additional research to determine what efficiencies or reduction of effort would actually result from the use of standardized ontologies for behavior representation.

Continued research is needed to help identify additional, appropriate military applications for the temporal "process" composability capability provided by the OWL behavior representation ontologies. This would be in addition to research on the standardization of behavior representation ontologies to further reuse of behaviors between and within simulations to reduce the KA/KE and VV&A efforts.

4 CONCLUSIONS

The following conclusions have been reached from the conduct of this research:

The explicit semantics for the behavior domain of Computer Generated Forces can be well represented using OWL ontologies,

The behavior representation ontologies, strongly patterned after the composable behavior paradigm of OOS, can definitely support composability,

Generalized RDF/XML behavior instances, conforming to the behavior ontologies, can be transformed using XSLT to the format used by a specific simulation to potentially support reuse of the composed behaviors across simulations.

The KA/KE information used to develop a behavior can be captured by the behavior metadata to potentially reduce the KA/KE effort when a behavior is reused in a simulation other than the one for which it was originally developed.

The VV&A process associated with the development of a specific behavior can be captured by the behavior metadata to potentially reduce the VV&A effort when a behavior is reused in a new composite behavior.

The metadata that may be associated with a composite or primitive behavior can aid in the filtered presentation of available behaviors through keyword searching or inferencing to potentially increase the efficiency with which behavior developers can compose new behaviors,

A simulation-independent standardization effort for behavior representation ontologies with a much greater diversity of simulations participating, along with one or more simulation-independent behavior composer tools for producing RDF/XML instances of behaviors that conform to the standardized ontologies, are needed before the potential benefits mentioned herein can be realized.

5 SUMMARY

DRC supported DMSO's Human Behavior Representation initiative by investigating the use of standard ontology behavior representations to support composability. DRC developed four generalized ontologies in the Web Ontology Language (OWL) for representing Computer Generated Force (CGF) behaviors. Those ontologies allowed for the composing of behavior instances in Resource Description Format (RDF) / Extensible Markup Language (XML) formats from primitive behaviors and other composite behaviors. The developed ontologies were:

- Behavior,
- Artifact,
- Variable, and
- ConceptDomainMetadata.

DRC developed a prototype behavior composer tool within the OOS simulation using the existing OOS graphical behavior composer tool. It was modified to allow transforming the XML behaviors created using the tool from the native OOS XML format into an RDF/XML format committed to the developed OWL behavior ontologies. The transformation was accomplished using Extensible Stylesheet Language Transformation (XSLT) files embedded within the prototype. The prototype tool also allowed the user to add metadata to the RDF/XML composed behavior instances that would represent the addition of Verification, Validation and Accreditation (VV&A) type data as well as data documenting the authoritative sources for the created behavior instances. The latter demonstrated the capturing of Knowledge Acquisition / Knowledge Engineering (KA/KE) information and tagging it to the developed behavior.

DRC added an additional capability, a simple filtering method that would allow a behavior developer to reduce the search space for the particular primitive or composite behavior needed during the composition. Currently, for example, an OOS behavior developer would have to either "brute-force" search through all the available behaviors (166 in the OOS Build 15), checking each for applicability, or have the name of the specific behaviors to use already known/memorized. With the filtering based on instance

data conforming to the ontologies, the captured RDF/XML instance data allowed the user to reduce the search space to only the behaviors that might be applicable, a much more manageable set.

The XSLT files for the ontologies were then used with the 166 OOS Build 15 behaviors formatted in XML to create instance files of those behaviors in RDF/XML formats compatible with the ontologies. They were then validated to confirm that they conformed to the ontologies.

For the JSAF option that was exercised, a couple of the behaviors from the vmove_task.fsm file were hand-coded, using the logic represented by the finite state machine (FSM) C-code, into RDF/XML instance files committed to the behavior ontologies. Those instance files were then validated to confirm that they conformed to the ontologies and that the JSAF behaviors could be suitably represented by them. To complete the loop, DRC developed another XSLT file that transformed the JSAF RDF/XML instance files back into the same "C-language" type of constructs in which they had originally appeared in the vmove_task.fsm file.

In a final ad hoc check to illustrate the capability of moving composed behaviors between disparate simulations, the JSAF XSLTs were modified and used on some of the OOS behaviors that had been transformed into the RDF/XML formats. The output looked similar to what one would expect in the JSAF finite state machine format.

6 REFERENCES

Anderson, R. & Henninger, A. (December 2-5, 2002). KA/KE Hybrid Document – Versatility for V&V and Software Development. Proceedings of the Interservice/ Industry Training, Simulation and Education Conference (I/ITSEC) 2002, Orlando, FL.

Berners-Lee,'T. (1999). Weaving the Web, San Francisco: Harper.

Bjorkman, E., Tyler, J., & Barry, P. (28 August 2001). Common Human Behavior Representation and Interchange System (CHRIS) After-Action Report. Retrieved August 28, 2001, from http://www.msiac.dmso.mil/hobm/workshop/CHRIS_AAR5.doc.

Chairman Joint Chiefs of Staff (CJCS). (1 July 2002). Universal Joint Task List, CJCSM 3500.04C.

DaCosta, B. (May 7-9, 2002). XML Support for OneSAF Objective System Behaviors. Proceedings of the 11th Computer Generated Forces and Behavioral Representation (CGF&BR) Conference, Orlando, FL.

Fineberg, M. (1995). A Comprehensive Taxonomy of Human Behaviors for Synthetic Forces. IDA Paper P-3155.

Fineberg, M. (1998). Towards a General Theory and Taxonomy of Behavior. CSERIAC Gateway, Vol. IX, No. 2, pp 6-8.

Foster, P. (November 26-29, 1997). Cognitive Modeling of Doctrinal Behaviors in Automated Units: Application of Behavior Definition Frames in WARSIM 2000. Proceedings of the Interservice/ Industry Training, Simulation and Education Conference (I/ITSEC) 2001, Orlando, FL.

Henderson, C. & Grainger, B. (December 2-5, 2002). Composable Behaviors in the OneSAF Objective System. Proceedings of the Interservice/ Industry Training, Simulation and Education Conference (I/ITSEC) 2002. Orlando, FL.

JCMMS UML Style Guide, Version 0.4, 1997.

Karr, C. & Holbrook, R. (May 11-13, 1999). Modeling Command and Control in WARSIM 2000. Proceedings of the 8th Computer Generated Forces and Behavioral Representation (CGF&BR) Conference. Orlando, FL.

Knowledge Integration Resource Center (KIRC). Retrieved February 13, 2004, from http://65.222.166.4/.

Lacy, L. (Fall 2000). Computer Generated Forces Behavior Representation and Reuse Using the eXtensible Markup Language (XML). Proceedings of the Fall 2000 Simulation Interoperability Workshop (SIW).

Lacy, L. (December 1-4, 1997). Conceptual Models for WARSIM 2000. Proceedings of the Interservice/ Industry Training, Simulation and Education Conference (I/ITSEC) 1997, Orlando, FL.

Lacy, L. & Henninger, A. (December 1-4, 2003). Developing Primitive Behavior Ontologies using the Ontology Web Language. Proceedings of the Interservice/ Industry Training, Simulation and Education Conference (I/ITSEC) 2003, Orlando, FL, pp 575-585.

Law, D. & Moerk, J. (Spring 2003). PSF: A Portable Scenario Format For CCTT. Proceedings of the Spring 2003 Simulation Interoperability Workshop (SIW).

McEnany, B. & Marshall, H. (1994). CCTT SAF Functional Analysis. Proceedings of the Fourth Computer Generated Forces and Behavioral Representation (CGF&BR) Conference.

Ourston, D., Blanchard, D., Chandler, E., Loh, E., & Marshall, H. (1995). From CIS to Software. Proceedings of the Fifth Computer Generated Forces and Behavioral Representation (CGF&BR) Conference.

Sagan, D. (Spring 2000). Conceptual Modeling Lessons Learned from WARSIM 2000. Proceedings of the Spring Simulation Interoperability Workshop (SIW).

W3C Press Release. (2004). Retrieved June 21, 2004, from http:/www.w3c.org/2004/01/sws-pressrelease .html.en/

Attachment 1–Ontology Design Document

Final Report - Ontology Design Diagram
for Developing Standard Ontological Behavior Representations to Support Composability

Contract Number F33615-03-C-6341

Prepared for
Lt. Benjamin Hartlage
AFRL/HECS
2698 G Street, Bldg 190, Area B
Wright-Patterson AFB, OH 45433-7604

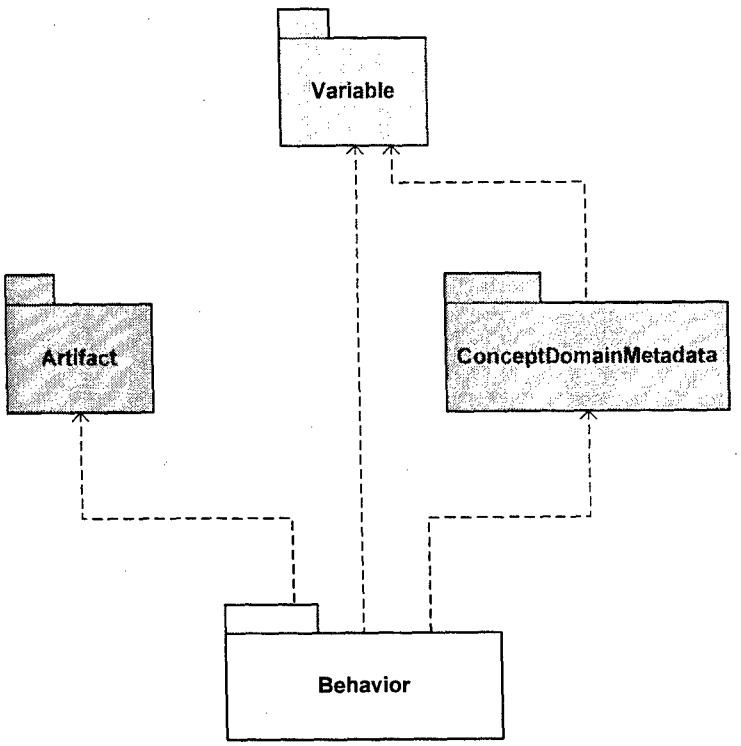
Prepared by
Dynamics Research Corporation
3505 Lake Lynda Drive, Suite 100
Orlando, FL 32817
(407) 380-1200 x 106

December 21, 2004

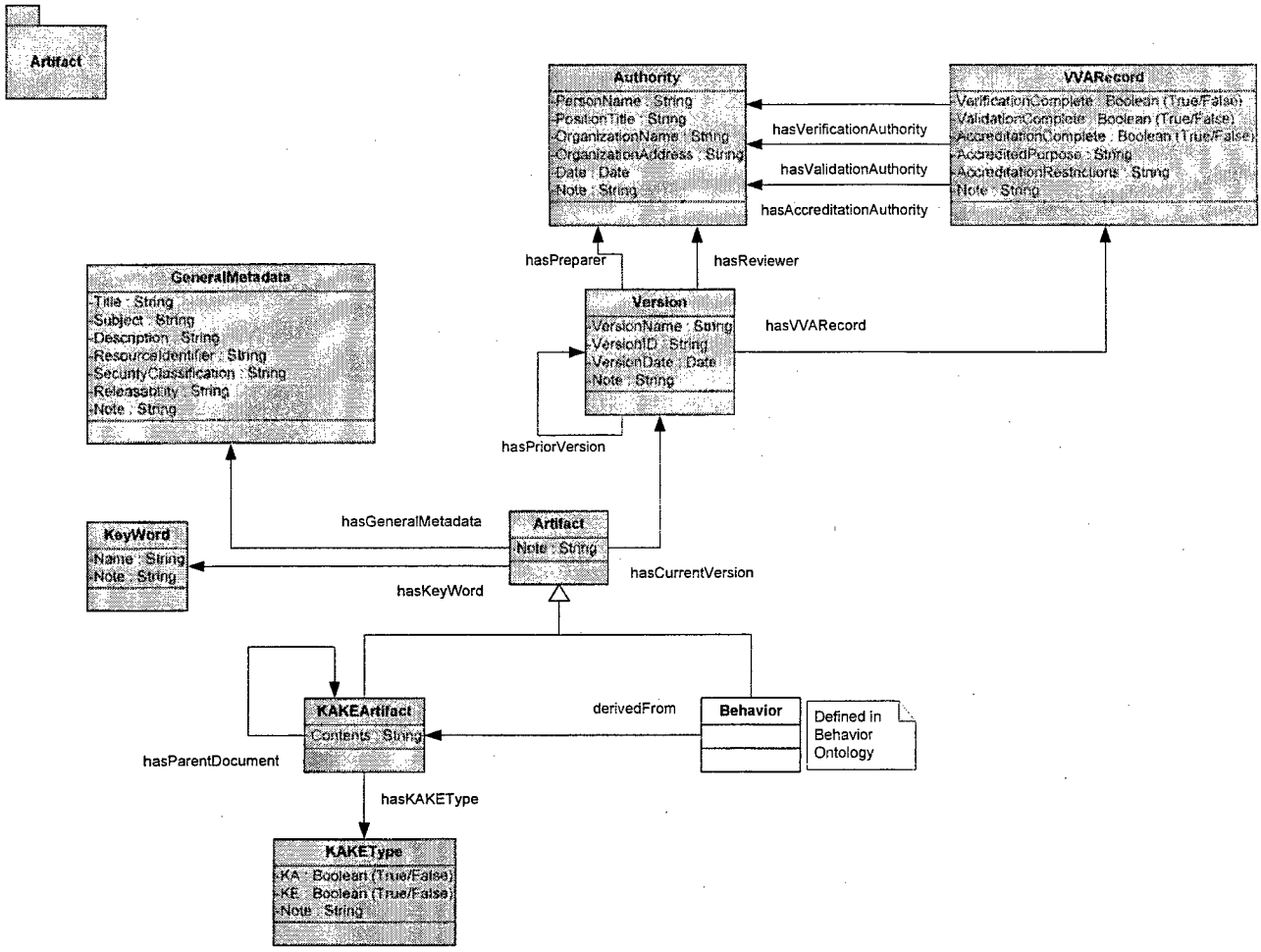
The following diagrams are based on the Software Requirements Specification
for Developing Standard Ontological Behavior Representations to Support Composability.

Table of Contents

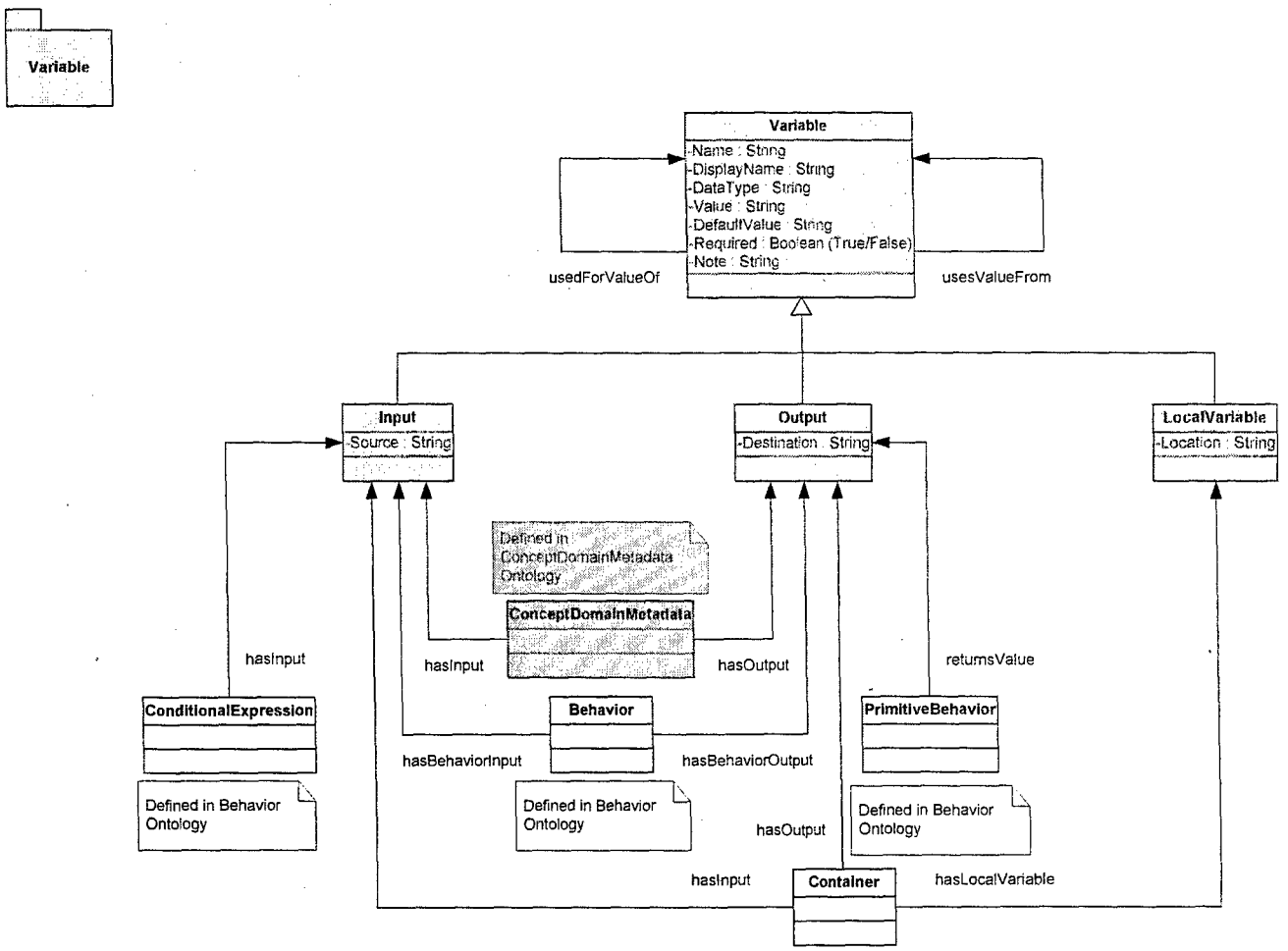
Imports Relationships Between Ontologies



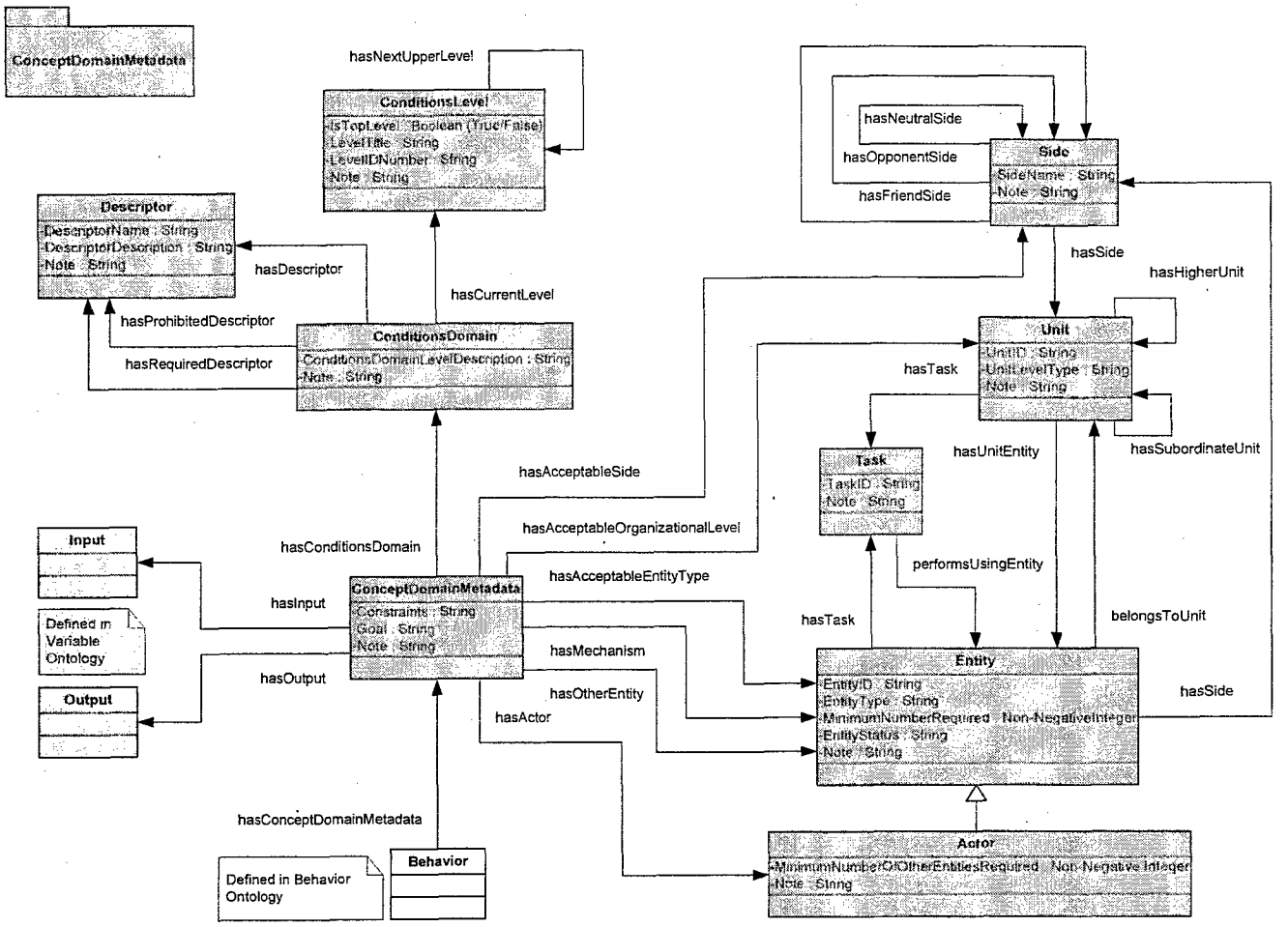
Page 1-3


Artifact
Authority
-PersonName : String
-PositionTitle : String
-OrganizationName : String
-OrganizationAddress : String
-Date : Date
-Note : String
VVARecord
-VerificationComplete : Boolean (True/False)
-ValidationComplete : Boolean (True/False)
-AccreditationComplete : Boolean (True/False)
-AccreditedPurpose : String
-AccreditationRestrictions : String
-Note : String
hasVerificationAuthority
hasValidationAuthority
hasAccreditationAuthority
hasPreparer
hasReviewer
GeneralMetadata
-Title : String
-Subject : String
-Description : String
-ResourceIdentifier : String
-SecurityClassification : String
-Releasability : String
-Note : String
Version
-VersionName : String
-VersionID : String
-VersionDate : Date
-Note : String
hasVVARecord
hasPriorVersion
hasGeneralMetadata
KeyWord
-Name : String
-Note : String
Artifact
-Note : String
hasCurrentVersion
hasKeyWord
KAKEArtifact
-Contents : String
derivedFrom
Behavior
Defined in Behavior Ontology
hasParentDocument
hasKAKEType
KAKEType
-KA : Boolean (True/False)
-KE : Boolean (True/False)
-Note : String

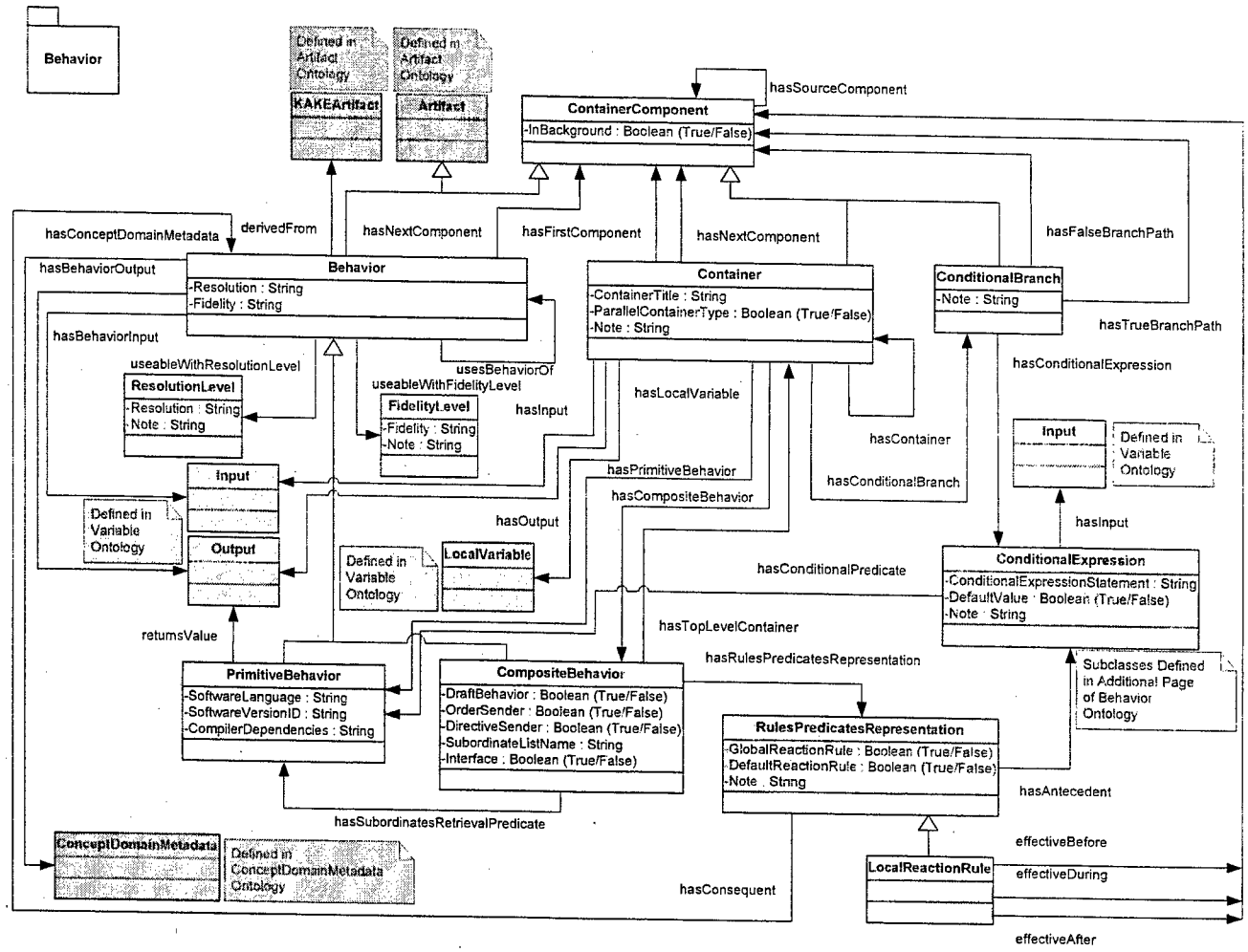
Page 1-4


Variable
Variable
-Name : String
-DisplayName : String
-DataType : String
-Value : String
-DefaultValue : String
-Required : Boolean (True/False)
-Note : String
usedForValueOf
usesValueFrom
Input
-Source : String
Output
-Destination : String
LocalVariable
-Location : String
Defined in ConceptDomainMetadata Ontology
ConceptDomainMetadata
hasInput
hasInput
hasOutput
returnsValue
ConditionalExpression
Behavior
PrimitiveBehavior
hasBehaviorInput
hasBehaviorOutput
Defined in Behavior Ontology
Defined in Behavior Ontology
Defined in Behavior Ontology
hasOutput
hasInput
Container
hasLocalVariable

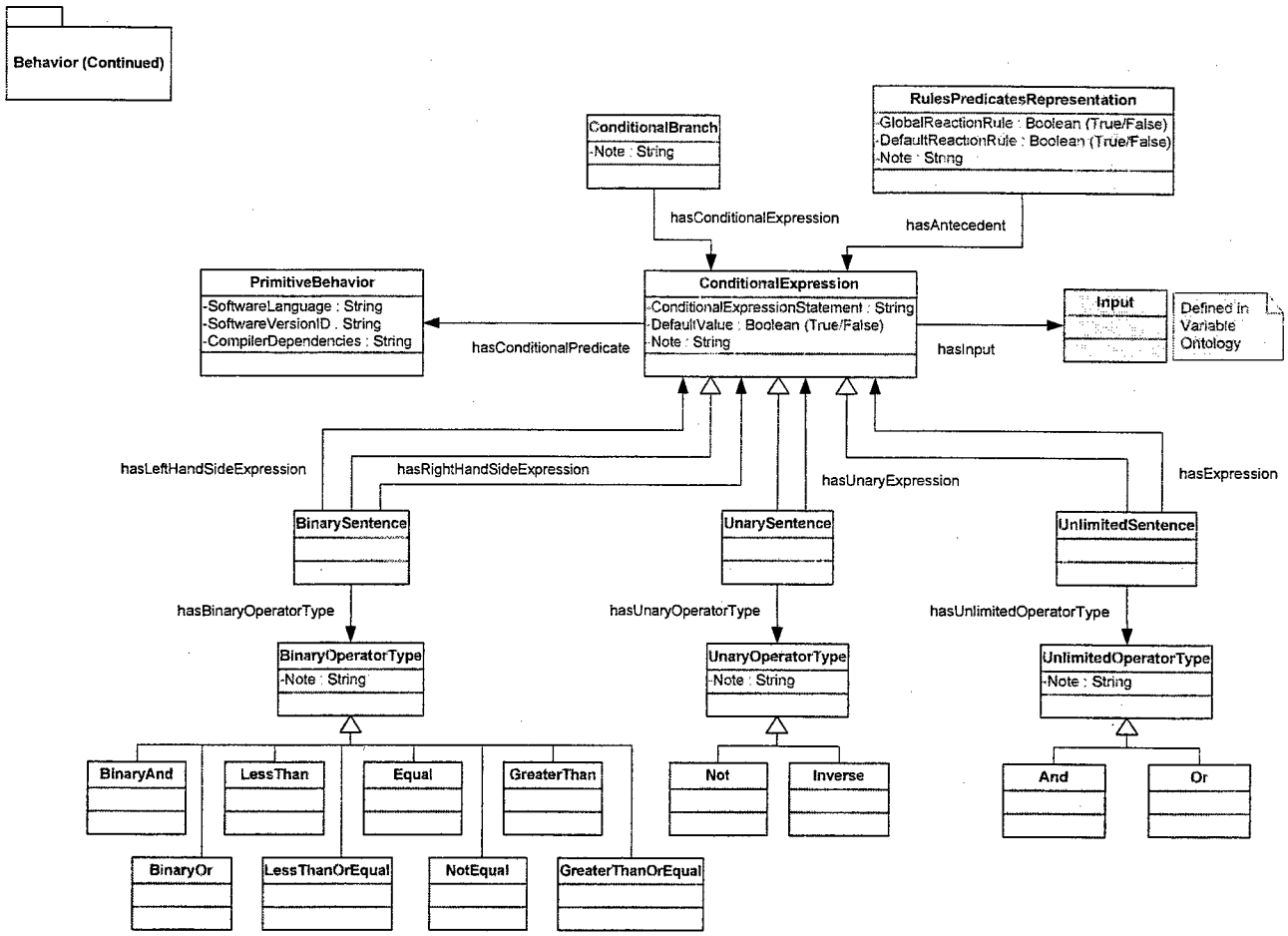
Page 1-5


ConceptDomainMetadata
hasNextUpperLevel
ConditionsLevel
isTopLevel : Boolean (True/False)
LevelTitle : String
LevelIDNumber : String
Note : String
hasNeutralSide
hasOpponentSide
hasFriendSide
Side
SideName : String
Note : String
hasSide
hasHigherUnit
Descriptor
DescriptorName : String
DescriptorDescription : String
Note : String
hasDescriptor
hasCurrentLevel
hasProhibitedDescriptor
hasRequiredDescriptor
ConditionsDomain
ConditionsDomainLevelDescription : String
Note : String
Unit
UnitID : String
UnitLevelType : String
Note : String
hasTask
hasUnitEntity
hasSubordinateUnit
Task
TaskID : String
Note : String
hasAcceptableSide
hasAcceptableOrganizationalLevel
hasAcceptableEntityType
performsUsingEntity
Input
hasConditionsDomain
hasInput
ConceptDomainMetadata
Constraints : String
Goal : String
Note : String
Defined in Variable Ontology
hasOutput
Output
hasMechanism
hasOtherEntity
hasActor
hasTask
belongsToUnit
Entity
EntityID : String
EntityType : String
MinimumNumberRequired : Non-NegativeInteger
EntityStatus : String
Note : String
hasSide
hasConceptDomainMetadata
Defined in Behavior Ontology
Behavior
Actor
MinimumNumberOfOtherEntitiesRequired : Non-NegativeInteger
Note : String

Page 1-6

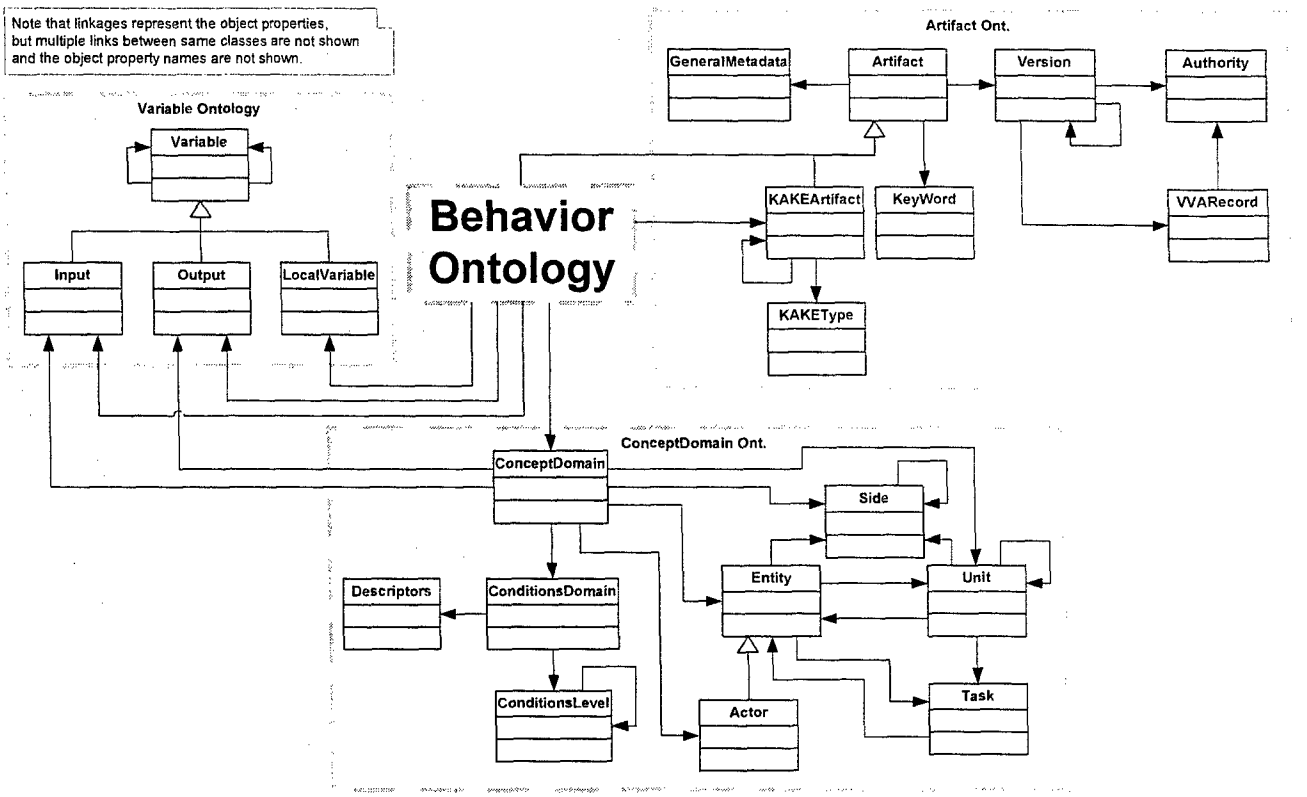

Behavior
Defined in Artifact Ontology
KAKEArtifact
Defined in Artifact Ontology
Artifact
ContainerComponent
-InBackground : Boolean (True/False)
hasSourceComponent
hasConceptDomainMetadata
derivedFrom
hasNextComponent
hasFirstComponent
hasNextComponent
hasFalseBranchPath
hasBehaviorOutput
Behavior
-Resolution : String
-Fidelity : String
Container
-ContainerTitle : String
-ParallelContainerType : Boolean (True/False)
-Note : String
ConditionalBranch
-Note : String
hasBehaviorInput
hasTrueBranchPath
useableWithResolutionLevel
usesBehaviorOf
hasConditionalExpression
ResolutionLevel
-Resolution : String
-Note : String
useableWithFidelityLevel
FidelityLevel
-Fidelity : String
-Note : String
hasInput
hasLocalVariable
hasContainer
Input
Defined in Variable Ontology
hasPrimitiveBehavior
hasConditionalBranch
Input
hasCompositeBehavior
Defined in Variable Ontology
hasOutput
hasInput
Output
LocalVariable
hasConditionalPredicate
ConditionalExpression
-ConditionalExpressionStatement : String
-DefaultValue : Boolean (True/False)
-Note : String
returnsValue
hasTopLevelContainer
hasRulesPredicatesRepresentation
Subclasses Defined in Additional Page of Behavior Ontology
PrimitiveBehavior
-SoftwareLanguage : String
-SoftwareVersionID : String
-CompilerDependencies : String
CompositeBehavior
-DraftBehavior : Boolean (True/False)
-OrderSender : Boolean (True/False)
-DirectiveSender : Boolean (True/False)
-SubordinateListName : String
-Interface : Boolean (True/False)
RulesPredicatesRepresentation
-GlobalReactionRule : Boolean (True/False)
-DefaultReactionRule : Boolean (True/False)
-Note : String
hasAntecedent
hasSubordinatesRetrievalPredicate
ConceptDomainMetadata
Defined in ConceptDomainMetadata Ontology
effectiveBefore
LocalReactionRule
effectiveDuring
hasConsequent
effectiveAfter

Page 1-7


Behavior (Continued)
ConditionalBranch
-Note : String
RulesPredicatesRepresentation
-GlobalReactionRule : Boolean (True/False)
-DefaultReactionRule : Boolean (True/False)
-Note : String
hasConditionalExpression
hasAntecedent
PrimitiveBehavior
-SoftwareLanguage : String
-SoftwareVersionID : String
-CompilerDependencies : String
ConditionalExpression
-ConditionalExpressionStatement : String
-DefaultValue : Boolean (True/False)
-Note : String
Input
Defined in Variable Ontology
hasConditionalPredicate
hasInput
hasLeftHandSideExpression
hasRightHandSideExpression
hasUnaryExpression
hasExpression
BinarySentence
UnarySentence
UnlimitedSentence
hasBinaryOperatorType
hasUnaryOperatorType
hasUnlimitedOperatorType
BinaryOperatorType
-Note : String
UnaryOperatorType
-Note : String
UnlimitedOperatorType
-Note : String
BinaryAnd
LessThan
Equal
GreaterThan
Not
Inverse
And
Or
BinaryOr
LessThanOrEqual
NotEqual
GreaterThanOrEqual

Page 1-8

Supporting Ontologies Summary



Attachment 2–Behavior Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xml:base="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#">
<owl:Ontology rdf:about="">
      <rdfs:comment>Classes for behavior composition--Behavior, ResolutionLevel,
FidelitLevel, PrimitiveBehavior, CompositeBehavior, Container,
ContainerComponent, ConditionalBranch, ConditionalExpression,
RulesPredicatesRepresentation, and LocalReactionRules.</rdfs:comment>
      <owl:imports
rdf:resource="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl"/>
      <owl:imports rdf:resource="http://orlando.drc.com/hbr/Ontologies/Artifact-
ont.owl"/>
      <owl:imports rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-
ont.owl"/>
      <rdfs:label>Behavior Ontology</rdfs:label>
</owl:Ontology>
<owl:Class rdf:ID="And">
     <rdfs:label>And</rdfs:label>
     <rdfs:comment>A subclass of an UnlimitedOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#UnlimitedOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="Behavior">
     <rdfs:label>Behavior</rdfs:label>
     <rdfs:comment>Superclass of PrimitiveBehavior and CompositeBehavior, the
actions which are performed by the Actor, and is a subclass of Artifact.  Has
datatype properties of ResolutionLevel and Fidelity and object type properties
of derivedFrom(KAKEArtifact), hasConceptDomainMetadata(ConceptDomainMetadata),
hasBehaviorOutput(Output), hasBehaviorInput(Input),
useableWithResolutionLevel(ResolutionLevel),
useableWithFidelityLevel(FidelityLevel), and
hasNextComponent(ContainerComponent).</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#ContainerComponent"/>
     <rdfs:subClassOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#Artifact"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Resolution"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#useableWithResolutionLevel"/>
                <owl:allValuesFrom rdf:resource="#ResolutionLevel" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Fidelity"/>
```

```
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#useableWithFidelityLevel"/>
                <owl:allValuesFrom rdf:resource="#FidelityLevel" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#usesBehaviorOf"/>
                <owl:allValuesFrom rdf:resource="#Behavior" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty
rdf:resource="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl#hasConceptDomainMetadata" />
                <owl:allValuesFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl#ConceptDomainMetadata" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Artifact-
ont.owl#derivedFrom" />
                <owl:allValuesFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Artifact-
ont.owl#KAKEArtifact" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasNextComponent" />
                <owl:allValuesFrom rdf:resource="#ContainerComponent" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-
ont.owl#hasBehaviorInput" />
                <owl:allValuesFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#Input" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-
ont.owl#hasBehaviorOutput" />
```

```
                <owl:allValuesFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#Output" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="BinaryAnd">
     <rdfs:label>BinaryAnd</rdfs:label>
     <rdfs:comment>A subclass of a BinaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#BinaryOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="BinaryOperatorType">
     <rdfs:label>BinaryOperatorType</rdfs:label>
     <rdfs:comment>The operator class for a BinarySentence</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="BinaryOr">
     <rdfs:label>BinaryOr</rdfs:label>
     <rdfs:comment>A subclass of a BinaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#BinaryOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="BinarySentence">
     <rdfs:label>BinarySentence</rdfs:label>
     <rdfs:comment>A subclass of ConditionalExpression, it allows for a left-
and right-hand-side ConditionalExpression with a
BinaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#ConditionalExpression"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasLeftHandSideExpression"/>
                <owl:allValuesFrom rdf:resource="#ConditionalExpression" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasRightHandSideExpression"/>
                <owl:allValuesFrom rdf:resource="#ConditionalExpression" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasBinaryOperatorType"/>
                <owl:allValuesFrom rdf:resource="#BinaryOperatorType" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="CompositeBehavior">
     <rdfs:label>CompositeBehavior</rdfs:label>
     <rdfs:comment>Behavior that is composed of one or more primitive behaviors
with temporal sequencing constructs, including possibly conditional branches,
that determine when the primitive behaviors execute.  The composite behavior can
only change the state of the simulation through execution of its included
```

```
primitive behaviors and/or the primitive behaviors included in other commposite
behaviors it may contain. It has datatype properties of DraftBehavior,
OrderSender, DirectiveSender, Interface, and SubordinateListName and object type
properties of hasTopLevelContainer(Container),
hasSubordinatesRetrievalPredicate(PrimitiveBehavior), and
hasRulesPredicatesRepresentation(RulesPredicatesRepresentation).</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#Behavior"/>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty
rdf:resource="#hasSubordinatesRetrievalPredicate" />
               <owl:allValuesFrom rdf:resource="#PrimitiveBehavior" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#hasTopLevelContainer"/>
               <owl:allValuesFrom rdf:resource="#Container" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#DraftBehavior"/>
               <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#SubordinateListName"/>
               <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#OrderSender"/>
               <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#DirectiveSender"/>
               <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#Interface"/>
               <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
```

```
                <owl:onProperty
rdf:resource="#hasRulesPredicatesRepresentation" />
                <owl:allValuesFrom
rdf:resource="#RulesPredicatesRepresentation" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ConditionalBranch">
     <rdfs:label>ConditionalBranch</rdfs:label>
     <rdfs:comment>A container component subclass and decision construct that
determines which of two paths are followed based on the evaluation of the
conditional expression.  It has a data type property of Note and object type
properties of hasTrueBranchPath(ContainerComponent),
hasFalseBranchPath(ContainerComponent), and
hasConditionalExpression(ConditionalExpression).</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#ContainerComponent"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConditionalExpression" />
                <owl:allValuesFrom rdf:resource="#ConditionalExpression" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasTrueBranchPath" />
                <owl:allValuesFrom rdf:resource="#ContainerComponent" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasFalseBranchPath" />
                <owl:allValuesFrom rdf:resource="#ContainerComponent" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ConditionalExpression">
     <rdfs:label>ConditionalExpression</rdfs:label>
     <rdfs:comment>It has a logical expression that can be evaluated and returns
a value that is either true or false or may reference a Predicate which returns
a value that is true or false.  It has data type properties of
conditionalExpression, DefaultValue and Note and object type properties of
hasInput(Input) and hasConditionalPredicate(PrimitiveBehavior).</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty
rdf:resource="#ConditionalExpressionStatement"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
```

```
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#DefaultValue"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#hasInput"
/>
                <owl:allValuesFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#Input" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Container">
     <rdfs:label>Container</rdfs:label>
     <rdfs:comment>An abstract construct, subclass of ContainerComponent, for
determining the temporal execution of container components, which can include
primitive behaviors, composite behaviors, conditional branches or other
containers.  Containers control the included components in either a parralel or
sequential order and in the background or foregrand..  It includes data type
properties of ContainerTitle, ParallelContainerType, and Note plus object type
properties of hasFirstComnponent(ContainerComponent),
hasNextComponent(ContainerComponent), hasPrimitiveBehavior(PrimititveBehavior),
hasCompositeBehavior(CompositeBehavior),
hasConditionalBranch(ConditionalBranch), hasInput(Input), hasOutput(Output), and
hasLocalVariable(LocalVariable).</rdfs:comment>

     <rdfs:subClassOf rdf:resource="#ContainerComponent"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#hasInput"
/>
                <owl:allValuesFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#Input" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#ContainerTitle"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
```

```
                <owl:onProperty rdf:resource="#ParallelContainerType"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasContainer"/>
                <owl:allValuesFrom rdf:resource="#Container" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasFirstComponent" />
                <owl:allValuesFrom rdf:resource="#ContainerComponent" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasNextComponent" />
                <owl:allValuesFrom rdf:resource="#ContainerComponent" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCompositeBehavior" />
                <owl:allValuesFrom rdf:resource="#CompositeBehavior" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasPrimitiveBehavior" />
                <owl:allValuesFrom rdf:resource="#PrimitiveBehavior" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConditionalBranch" />
                <owl:allValuesFrom rdf:resource="#ConditionalBranch" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#hasOutput"
/>
                <owl:allValuesFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#Output" />
          </owl:Restriction>
     </rdfs:subClassOf>
```

```
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-
ont.owl#hasLocalVariable" />
                <owl:allValuesFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-
ont.owl#LocalVariable" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ContainerComponent">
     <rdfs:label>ContainerComponent</rdfs:label>
     <rdfs:comment>A superclass that includes subclasses for Behavior
(PrimitiveBehavior and CompositeBehavior are further subclasses),
ConditionalBranch, and Container.  It has a datatype property of InBackground
and an object type property of hasSourceComponent.</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#InBackground"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasSourceComponent" />
                <owl:allValuesFrom rdf:resource="#ContainerComponent" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Equal">
     <rdfs:label>Equal</rdfs:label>
     <rdfs:comment>A subclass of a BinaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#BinaryOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="FidelityLevel">
     <rdfs:label>FidelityLevel</rdfs:label>
     <rdfs:comment>Captures the Fidelity levels with which the Behavior could be
used.  It includes data type properties of Name and Note.</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Fidelity"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="GreaterThan">
     <rdfs:label>GreaterThan</rdfs:label>
```

```
     <rdfs:comment>A subclass of a BinaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#BinaryOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="GreaterThanOrEqual">
     <rdfs:label>GreaterThanOrEqual</rdfs:label>
     <rdfs:comment>A subclass of a BinaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#BinaryOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="Inverse">
     <rdfs:label>Inverse</rdfs:label>
     <rdfs:comment>A subclass of a UnaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#UnaryOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="LessThan">
     <rdfs:label>LessThan</rdfs:label>
     <rdfs:comment>A subclass of a BinaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#BinaryOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="LessThanOrEqual">
     <rdfs:label>LessThanOrEqual</rdfs:label>
     <rdfs:comment>A subclass of a BinaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#BinaryOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="LocalReactionRule">
     <rdfs:label>LocalReactionRule</rdfs:label>
     <rdfs:comment>Subclass of the RulesPredicatesRepresentation class.  It
points to sub-behaviors of the composite behavior during which the rule or
predicate would be effective for interrupting the composite behavior execution.
It has an object type property of
effectiveDuring(ContainerComponent).</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#RulesPredicatesRepresentation"/>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#effectiveDuring"/>
               <owl:allValuesFrom rdf:resource="#ContainerComponent" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#effectiveAfter"/>
               <owl:allValuesFrom rdf:resource="#ContainerComponent" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#effectiveBefore"/>
               <owl:allValuesFrom rdf:resource="#ContainerComponent" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Not">
     <rdfs:label>Not</rdfs:label>
     <rdfs:comment>A subclass of a UnaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#UnaryOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="NotEqual">
     <rdfs:label>NotEqual</rdfs:label>
```

```
     <rdfs:comment>A subclass of a BinaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#BinaryOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="Or">
     <rdfs:label>Or</rdfs:label>
     <rdfs:comment>A subclass of an UnlimitedOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#UnlimitedOperatorType"/>
</owl:Class>
<owl:Class rdf:ID="PrimitiveBehavior">
     <rdfs:label>PrimitiveBehavior</rdfs:label>
     <rdfs:comment>A Behavior subclass that is basic and encoded in the software
of the simulation.  It can cause the state of the simulation to change.  Class
includes its resolution and fidelity levels and matching levels for both
inherited form Behavior superclass and includes data type properties of
SoftwareLanguage and SoftwareVersionID and CompilerDependencies within the class
directly.</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#Behavior"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#SoftwareLanguage"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#SoftwareVersionID"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#CompilerDependencies"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ResolutionLevel">
     <rdfs:label>ResolutionLevel</rdfs:label>
     <rdfs:comment>Captures the ResolutionLevels with which the Behavior could
be used.  It includes data type properties of Name and Note.</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Resolution"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
```

```
</owl:Class>
<owl:Class rdf:ID="RulesPredicatesRepresentation">
     <rdfs:label>RulesPredicatesRepresentation</rdfs:label>
     <rdfs:comment>A construct that is used to interrupt a behavior if its
antecedent (conditional expression) is true.  The interruption causes the
consequent (a behavior) to execute.  It includes boolean values for whether the
rule/predicate is a global reaction rule or default reaction rule.  It has data
type properties of GlobalReactionRule, DefaultReactionRule, and Note plus object
type properties of hasAntecedant(ConditionalExpression) and
hasConsequent(Behavior).  It has a LocalReactionRule subclass that points to the
ContainerComponent(s) for which it is effective.</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#GlobalReactionRule"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#DefaultReactionRule"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasAntecedent" />
                <owl:allValuesFrom rdf:resource="#ConditionalExpression" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConsequent" />
                <owl:allValuesFrom rdf:resource="#Behavior" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="UnaryOperatorType">
     <rdfs:label>UnaryOperatorType</rdfs:label>
     <rdfs:comment>The operator class for a UnarySentence</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="UnarySentence">
```

```
     <rdfs:label>UnarySentence</rdfs:label>
     <rdfs:comment>A subclass of ConditionalExpression, it allows for a
ConditionalExpression with a UnaryOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#ConditionalExpression"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnaryExpression"/>
                <owl:allValuesFrom rdf:resource="#ConditionalExpression" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnaryOperatorType"/>
                <owl:allValuesFrom rdf:resource="#UnaryOperatorType" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="UnlimitedOperatorType">
     <rdfs:label>UnlimitedOperatorType</rdfs:label>
     <rdfs:comment>The operator class for an UnlimitedSentence</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="UnlimitedSentence">
     <rdfs:label>UnlimitedSentence</rdfs:label>
     <rdfs:comment>A subclass of ConditionalExpression, it allows for one or
more ConditionalExpressions with an UnlimitedOperatorType</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#ConditionalExpression"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnlimitedOperatorType"/>
                <owl:allValuesFrom rdf:resource="#UnlimitedOperatorType" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf;resource="#hasExpression"/>
                <owl:allValuesFrom rdf:resource="#ConditionalExpression" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:DatatypeProperty rdf:ID="Resolution">
     <rdfs:label>Resolution</rdfs:label>
     <rdfs:comment>Level of resolution either of the behavior or resolution
level which could be used with the behavior..</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="useableWithResolutionLevel">
     <rdfs:label>useableWithResolutionLevel</rdfs:label>
     <rdfs:comment>Points to a resolution level which could be used with the
behavior.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="Fidelity">
```

```
    <rdfs:label>Fidelity</rdfs:label>
    <rdfs:comment>Fidelity level of the behavior or fideliity level which could
be used with the behavior.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="useableWithFidelityLevel">
    <rdfs:label>useableWithFidelityLevel</rdfs:label>
    <rdfs:comment>Points to a fidelity level which could be used with the
behavior.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="Note">
    <rdfs:label>Note</rdfs:label>
    <rdfs:comment>A human readable note.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasPrimitiveBehavior">
    <rdfs:label>hasPrimitiveBehavior</rdfs:label>
    <rdfs:comment>Points to a PrimitiveBehavior class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCompositeBehavior">
    <rdfs:label>hasCompositeBehavior</rdfs:label>
    <rdfs:comment>Points to a CompositeBehavior class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSubordinatesRetrievalPredicate">
    <rdfs:label>hasSubordinatesRetrievalPredicate</rdfs:label>
    <rdfs:comment>Points to a PrimitiveBehaviors class that is used as a
function to return the list of subordinates to the Actor conducting the
CompositeBehavior.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="SoftwareLanguage">
    <rdfs:label>SoftwareLanguage</rdfs:label>
    <rdfs:comment>Name of the software language used to implement the primitive
behavior.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="SoftwareVersionID">
    <rdfs:label>SoftwareVersionID</rdfs:label>
    <rdfs:comment>Version Identification of the software language used to
implement the primitive behavior.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="CompilerDependencies">
    <rdfs:label>CompilerDependencies</rdfs:label>
    <rdfs:comment>Description of acceptable compilers for the primitive
behavior software.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DraftBehavior">
    <rdfs:label>DraftBehavior</rdfs:label>
    <rdfs:comment>A boolean of true if the behavior is a draft behavior or
false if it is a completed composite behavior.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasNextComponent">
    <rdfs:label>hasNextComponent</rdfs:label>
    <rdfs:comment>Points to a ContainerComponent class, which is the next
portion of the behavior to execute at the completion of the Behavior or
Container that is pointing to it.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSourceComponent">
    <rdfs:label>hasSourceComponent</rdfs:label>
    <rdfs:comment>Points to a ContainerComponent class that precedes the
current ContainerComponent in execution.</rdfs:comment>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasTrueBranchPath">
     <rdfs:label>hasTrueBranchPath</rdfs:label>
     <rdfs:comment>Points to a ContainerComponent class that follows the
ControlBranch class if the returned value of the ConditionalExpression is
True.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasFalseBranchPath">
     <rdfs:label>hasFalseBranchPath</rdfs:label>
     <rdfs:comment>Points to a ContainerComponent class that follows the
ControlBranch class if the returned value of the ConditionalExpression is
False.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasConsequent">
     <rdfs:label>hasConsequent</rdfs:label>
     <rdfs:comment>Points to a Behavior class that would execute if the
RulesPredicatesRepresentation class' ConditionalExpression returns a value of
True during a portion of the Behavior when the RulesPredicatesRepresentation
class is effective.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasFirstComponent">
     <rdfs:label>hasFirstComponent</rdfs:label>
     <rdfs:comment>Points to a ContainerComponent class from a Container that is
a Sequence Container.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="ContainerTitle">
     <rdfs:label>ContainerTitle</rdfs:label>
     <rdfs:comment>Title of the Container.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="ParallelContainerType">
     <rdfs:label>ParallelContainerType</rdfs:label>
     <rdfs:comment>Boolean value of True if the Container is a Parallel
Container or False if the Container is a Sequence Container.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="InBackground">
     <rdfs:label>InBackground</rdfs:label>
     <rdfs:comment>Boolean of True if the ContainerComponent is to operate in
the background or False if it is to operate in the foreground.  If in the
background, the ContainerComponent is considered to have finished executing as
soon as it is started so that no following executing ContainerComponent waits
for the ContainerComponent to actually finish executing.  If in the foreground,
the ContainerComponent must finish before any following ContainerComponent can
begin executing.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasContainer">
     <rdfs:label>hasContainer</rdfs:label>
     <rdfs:comment>Points to a Container class from a Container.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasConditionalBranch">
     <rdfs:label>hasConditionalBranch</rdfs:label>
     <rdfs:comment>Points to a ConditionalBranch class from a
Container.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasConditionalExpression">
     <rdfs:label>hasConditionalExpression</rdfs:label>
     <rdfs:comment>Points to a ConditionalExpression class from a
ConditionalBranch class.</rdfs:comment>
```

```
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="ConditionalExpressionStatement">
    <rdfs:label>ConditionalExpressionStatement</rdfs:label>
    <rdfs:comment>An arbitrarily complex logical expression that may include
predicates as well as "and", "or", or "not" connectives, the value of which
determines which of two paths is to be followed to the next component to be
executed, or whether or not a transitional rule will be triggered to end the
behavior..</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DefaultValue">
    <rdfs:label>DefaultValue</rdfs:label>
    <rdfs:comment>Value of the Conditional Expression after it has finished
being evaluated if there is no value that can be returned from either a
ConditttionalExpressionStatement or a ConditionalPredicate.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasRulesPredicatesRepresentation">
    <rdfs:label>hasRulesPredicatesRepresentation</rdfs:label>
    <rdfs:comment>Points to a RulesPredicatesRepresentation
class.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="GlobalReactionRule">
    <rdfs:label>GlobalReactionRule</rdfs:label>
    <rdfs:comment>A reaction rule that applies throughout the entire execution
of a composite behavior.  Boolean value of True or False indicaties whether or
not the reaction rule is applicable globally during the behavior
execution.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DefaultReactionRule">
    <rdfs:label>DefaultReactionRule</rdfs:label>
    <rdfs:comment>The Global or Local Reaction Rules of a composite or
primitive behavior included in a higher level composite behavior, all of which
apply to the higher level composite behavior during the period that the included
behavior is executed.  Boolean value of True or False indicaties whether or not
the reaction rule is applicable.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasAntecedent">
    <rdfs:label>hasAntecedent</rdfs:label>
    <rdfs:comment>Has a conditional expression, the evaluation of which
determines if the rule or predicate is true or false and will terminate the
current behavior.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasTopLevelContainer">
    <rdfs:label>hasTopLevelContainer</rdfs:label>
    <rdfs:comment>Points to the Container class at the top level of a composite
behavior.</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="effectiveDuring">
    <rdfs:label>effectiveDuring</rdfs:label>
    <rdfs:comment>Points to a container component during which the Local
Reaction Rule is applicable.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasConditionalPredicate">
    <rdfs:label>hasConditionalPredicate</rdfs:label>
    <rdfs:comment>Points to a PrimitiveBehavior class that acts as a function
to return a True or False value.</rdfs:comment>
```

```
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="OrderSender">
     <rdfs:label>OrderSender</rdfs:label>
     <rdfs:comment>Booolean, where true means the composite behavior is an Order
Sender, I.e., the behavior is sent to subordinates to execute.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DirectiveSender">
     <rdfs:label>DirectiveSender</rdfs:label>
     <rdfs:comment>Booolean, where true means the composite behavior is a
Directive Sender, I.e., the behavior is sent to subordinates to direct
them.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Interface">
     <rdfs:label>Interface</rdfs:label>
     <rdfs:comment>Booolean, where true means the composite is controlled by an
interface to a user who provides the inputs for execution.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="SubordinateListName">
     <rdfs:label>SubordinateListName</rdfs:label>
     <rdfs:comment>Name of the list of subordinates for receipt of orders or
directives</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="usesBehaviorOf">
     <rdfs:label>usesBehaviorOf</rdfs:label>
     <rdfs:comment>Points to Behavior class</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasUnaryExpression">
     <rdfs:label>hasUnaryExpression</rdfs:label>
     <rdfs:comment>Points to the conditional expression for the unary
operator</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasLeftHandSideExpression">
     <rdfs:label>hasLeftHandSideExpression</rdfs:label>
     <rdfs:comment>Points to the left-hand-side conditional expression for a
binary operator</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasRightHandSideExpression">
     <rdfs:label>hasRightHandSideExpression</rdfs:label>
     <rdfs:comment>Points to the right-hand-side conditional expression for a
binary operator</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasUnaryOperatorType">
     <rdfs:label>hasUnaryOperatorType</rdfs:label>
     <rdfs:comment>Points to the UnaryOperatorType class</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasBinaryOperatorType">
     <rdfs:label>hasBinaryOperatorType</rdfs:label>
     <rdfs:comment>Points to the BinaryOperatorType class</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasUnlimitedOperatorType">
     <rdfs:label>hasUnlimitedOperatorType</rdfs:label>
     <rdfs:comment>Points to the UnlimitedOperatorType class</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasExpression">
     <rdfs:label>hasExpression</rdfs:label>
```

```
    <rdfs:comment>Points to a ConditionalExpression class from an
UnlimitedSentence</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="effectiveBefore">
    <rdfs:label>effectiveBefore</rdfs:label>
    <rdfs:comment>Points to a container component before which the Local
Reaction Rule is applicable.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="effectiveAfter">
    <rdfs:label>effectiveAfter</rdfs:label>
    <rdfs:comment>Points to a container component after which the Local
Reaction Rule is applicable.</rdfs:comment>
</owl:ObjectProperty>
</rdf:RDF>
```

Attachment 3 - Variable Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:var="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#"
      xml:base="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#">
<owl:Ontology rdf:about="">
      <rdfs:comment>Provides for Input, Output and Local Variable classes as
subclasses of the Variable class.</rdfs:comment>
      <rdfs:label>Variable Ontology</rdfs:label>
</owl:Ontology>
<owl:Class rdf:ID="Input">
     <rdfs:label>Input</rdfs:label>
     <rdfs:comment>A subclass of Variable,  it includes name, display name, data
type, value, a human-readable notation, and information on whether the input is
mandatory or not, all from Variable, as well as its own data type property of
Source.</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#Variable"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Source"/>
                <owl:allValuesFrom
 rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="LocalVariable">
     <rdfs:label>LocalVariable</rdfs:label>
     <rdfs:comment>A subclass of Variable,  it includes name, display name, data
 type, value, a human-readable notation, and information on whether the input is
 mandatory or not, all from Variable, as well as its own data type property of
 Location.</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#Variable"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Location"/>
                <owl:allValuesFrom
 rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Output">
     <rdfs:label>Ouput</rdfs:label>
     <rdfs:comment>A subclass of Variable,  it includes name, display name, data
 type, value, a human-readable notation, and information on whether the input is
 mandatory or not, all from Variable, as well as its own data type property of
 Destination.</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#Variable"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Destination"/>
                <owl:allValuesFrom
 rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
```

```
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Variable">
    <rdfs:label>Variable</rdfs:label>
    <rdfs:comment>A superclass that has data type properties of Name,
DisplayName, DataType, Value, Required, and Note and object type property of
usesValueFrom(Variable).  It has subclasses of Input, Output, and
LocalVariable.</rdfs:comment>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#Value"/>
            <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#Required"/>
            <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#DisplayName"/>
            <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#usesValueFrom"/>
            <owl:allValuesFrom rdf:resource="#Variable" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#DefaultValue"/>
            <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#usedForValueOf"/>
            <owl:allValuesFrom rdf:resource="#Variable" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#Name"/>
            <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
```

```
                <owl:onProperty rdf:resource="#DataType"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasInput">
     <rdfs:label>hasInput</rdfs:label>
     <rdfs:comment>Points to an Input.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasOutput">
     <rdfs:label>hasOutput</rdfs:label>
     <rdfs:comment>Points to an Output.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasLocalVariable">
     <rdfs:label>hasLocalVariable</rdfs:label>
     <rdfs:comment>Points to a Local Variable.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="Source">
     <rdfs:label>Source</rdfs:label>
     <rdfs:comment>A resource that provides the input.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Required">
     <rdfs:label>Required</rdfs:label>
     <rdfs:comment>Boolean expression indicating whether or not the object is
required.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Note">
     <rdfs:label>Note</rdfs:label>
     <rdfs:comment>A human-readable notation.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Destination">
     <rdfs:label>Destination</rdfs:label>
     <rdfs:comment>A resource that receives the output.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Location">
     <rdfs:label>Location</rdfs:label>
     <rdfs:comment>A resource that provides and/or receives the local
variable.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Name">
     <rdfs:label>Name</rdfs:label>
     <rdfs:comment>Name of the variable.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DataType">
     <rdfs:label>DataType</rdfs:label>
     <rdfs:comment>Datatype of the variable.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:ID="Value">
     <rdfs:label>Value</rdfs:label>
     <rdfs:comment>Value of the variable.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasBehaviorInput">
     <rdfs:label>hasBehaviorInput</rdfs:label>
     <rdfs:comment>Points to Input class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasBehaviorOutput">
     <rdfs:label>hasBehaviorOutput</rdfs:label>
     <rdfs:comment>Points to Output class.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="DisplayName">
     <rdfs:label>DisplayName</rdfs:label>
     <rdfs:comment>Name to be displayed to human user</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="usesValueFrom">
     <rdfs:label>usesValueFrom</rdfs:label>
     <rdfs:comment>Points to a Variable class</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="DefaultValue">
     <rdfs:label>DefaultValue</rdfs:label>
     <rdfs:comment>Default value for the variable if a value isn't
specified</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="usedForValueOf">
     <rdfs:label>usedForValueOf</rdfs:label>
     <rdfs:comment>Points to Variable class</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="returnsValue">
     <rdfs:label>returnsValue</rdfs:label>
     <rdfs:comment>Points to Output class for value returned by a
function</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
</rdf:RDF>
```

Attachment 4–Artifact Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xml:base="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#">
<owl:Ontology rdf:about="">
      <rdfs:comment>Provides Artifact superclass for Behavior and the KA/KE
Artifact basis for the behavior. Includes Artifact, GeneralMetadata, Version,
KAKEArtifact, KeyWord, KAKEType, Authority, and VVARecord
classes.</rdfs:comment>
      <rdfs:label>Artifact Ontology</rdfs:label>
</owl:Ontology>
<owl:Class rdf:ID="Artifact">
     <rdfs:label>Artifact</rdfs:label>
     <rdfs:comment>Superclass for Behavior and KAKEArtifact; includes datatype
property of Note and object type propertyies of
hasGeneralMetadata(GeneralMetadata) and
hasCurrentVersion(Version).</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasKeyWord"/>
                <owl:allValuesFrom rdf:resource="#KeyWord" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasGeneralMetadata"/>
                <owl:allValuesFrom rdf:resource="#GeneralMetadata" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCurrentVersion"/>
                <owl:allValuesFrom rdf:resource="#Version" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Authority">
     <rdfs:label>Authority</rdfs:label>
     <rdfs:comment>Includes person's name, position title, organization name,
organization address, date and a human-readable notation.</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#PersonName"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
```

```
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#PositionTitle"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#OrganizationName"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#OrganizationAddress"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#Date"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#date" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
    <owl:Class rdf:ID="GeneralMetadata">
        <rdfs:label>GeneralMetadata</rdfs:label>
        <rdfs:comment>Includes general metadata about the artifact (KAKEArtifact or
    Behavior), such as title, subject, description, resource identifier, security
    classification, releasability, and a note.  It includes data type properties of
    Title, Subject, Description, ResourceIdentifier, SecurityClassification,
    Releasability, and Note.  Version data is stored separately in
    Version.</rdfs:comment>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#Title"/>
                <owl:allValuesFrom
    rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#Subject"/>
```

```
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Releasability"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Description"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#ResourceIdentifier"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#SecurityClassification"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="KAKEArtifact">
     <rdfs:label>KAKEArtifact</rdfs:label>
     <rdfs:comment>A subclass of Artifact, it provides information about the
documentation for the KA/KEArtifact used as the basis for the behavior that is
derived from it.  Includes data type property of Contents and object type
properties of hasKAKEType(KAKEType), hasKeyWord(KeyWord), and hasParent
Document(KAKEArtifact).</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#Artifact"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Contents"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
```

```
                <owl:onProperty rdf:resource="#hasKAKEType" />
                <owl:allValuesFrom rdf:resource="#KAKEType" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasParentDocument"/>
                <owl:allValuesFrom rdf:resource="#KAKEArtifact" />
            </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="KAKEType">
        <rdfs:label>KAKEType</rdfs:label>
        <rdfs:comment>Describes the type of KAKEArtifact using Boolean values for
its data type properties of KA and KE, allowing designation of whether the type
is KA, KE, both KA and KE, or neither.</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="KeyWord">
        <rdfs:label>KeyWord</rdfs:label>
        <rdfs:comment>Provides the location for storing key words about the
KAKEArtifact.  It has data type properties of Name and Note.</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="Version">
        <rdfs:label>Version</rdfs:label>
        <rdfs:comment>A class that includes the revision history for a Behavior's
class or a KAKEArtifact's class.  It includes data type properties for  the
VersionName, VersionID, VersionDate, and Note.  It also has object type
properties of  hasPreparer(Authority), hasReviewer(Authority),
hasVVARecord(VVARecord), and hasPriorVersion(Version).</rdfs:comment>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#VersionID"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#VersionDate"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#date" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#VersionName"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasPriorVersion"/>
                <owl:allValuesFrom rdf:resource="#Version" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
```

```
            <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasVVARecord"/>
                  <owl:allValuesFrom rdf:resource="#VVARecord" />
            </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:onProperty rdf:resource="#Note"/>
                  <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasPreparer"/>
                  <owl:allValuesFrom rdf:resource="#Authority" />
            </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasReviewer"/>
                  <owl:allValuesFrom rdf:resource="#Authority" />
            </owl:Restriction>
      </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="VVARecord">
      <rdfs:label>VerificationValidationAccreditationRecord</rdfs:label>
      <rdfs:comment>Provides booleans denoting whether verification, validation
and accreditation are complete, and the accredited purpose,  accreditation
restrictions, and a note. (Data type properties of VerificationComplete,
ValidationComplete, AccreditationComplete,AccreditedPurpose,
AccreditationRestrictions, and Note.)  It also has object type properties for
hasVerification(Authority), hasValidationAuthority(Authority) and
hasAccreditationAuthority(Authority).</rdfs:comment>
      <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:onProperty rdf:resource="#VerificationComplete"/>
                  <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
            </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:onProperty rdf:resource="#ValidationComplete"/>
                  <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
            </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <owl:Restriction>
                  <owl:onProperty rdf:resource="#AccreditationComplete"/>
                  <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
            </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <owl:Restriction>
```

```
                <owl:onProperty rdf:resource="#AccreditedPurpose"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#AccreditationRestrictions"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasVerificationAuthority"/>
                <owl:allValuesFrom rdf:resource="#Authority" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasValidationAuthority"/>
                <owl:allValuesFrom rdf:resource="#Authority" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasAccreditationAuthority"/>
                <owl:allValuesFrom rdf:resource="#Authority" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:ObjectProperty rdf:ID="derivedFrom">
     <rdfs:label>derivedFrom</rdfs:label>
     <rdfs:comment>Points to the KAKEArtifact class.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="Contents">
     <rdfs:label>Contents</rdfs:label>
     <rdfs:comment>Provides a description of the information contained in the
KA/KE artifact.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="KAType">
     <rdfs:label>KAType</rdfs:label>
     <rdfs:comment>Boolean attribute for KAKEArtifact that is True if the
document is of the KA type only or combined KA/KE type only and False otherwise,
e.g., is of another type than strictly KA or strictly KA and KE
only.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="KeywordName">
     <rdfs:label>KeywordName</rdfs:label>
     <rdfs:comment>List of key words.</rdfs:comment>
```

```
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Note">
     <rdfs:label>Note</rdfs:label>
     <rdfs:comment>A human readable note.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="VersionID">
     <rdfs:label>VersionID</rdfs:label>
     <rdfs:comment>Version Identification of a version of a KA/KE
artifact.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="VersionDate">
     <rdfs:label>VersionDate</rdfs:label>
     <rdfs:comment>Date of a version of a KA/KE artifact.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasVVARecord">
     <rdfs:label>hasVVARecord</rdfs:label>
     <rdfs:comment>Points to a VVARecord class.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="VerificationComplete">
     <rdfs:label>VerificationComplete</rdfs:label>
     <rdfs:comment>Boolean, where True if the verification is complete or False
if the verification is not complete.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="ValidationComplete">
     <rdfs:label>ValidationComplete</rdfs:label>
     <rdfs:comment>Boolean, where True if the validation is complete or False if
the validation is not complete.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="AccreditationComplete">
     <rdfs:label>AccreditationComplete</rdfs:label>
     <rdfs:comment>Boolean, where True if the accreditation is complete or False
if the accreditation is not complete.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="AccreditedPurpose">
     <rdfs:label>AccreditedPurpose</rdfs:label>
     <rdfs:comment>A description of the approved purpose for use of the
behavior.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="AccreditationRestrictions">
     <rdfs:label>AccreditationRestrictions</rdfs:label>
     <rdfs:comment>A description of the restrictions, if any, on the use of the
behavior.  If full accreditation, this would be "None."  If limited
accreditation, this wouild be a description of the constraints.  If modification
of the simulation or additional information is needed before accreditation, this
would be stated.  Also, if accreditation is denied, this would be reflected as
well.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasVerificationAuthority">
     <rdfs:label>hasVerificationAuthority</rdfs:label>
     <rdfs:comment>Points to Authority class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasValidationAuthority">
     <rdfs:label>hasValidationAuthority</rdfs:label>
     <rdfs:comment>Points to Authority class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasAccreditationAuthority">
     <rdfs:label>hasAccreditationAuthority</rdfs:label>
```

```
    <rdfs:comment>Points to Authority class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPreparer">
    <rdfs:label>hasPreparer</rdfs:label>
    <rdfs:comment>Points to Authority class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasReviewer">
    <rdfs:label>hasReviewer</rdfs:label>
    <rdfs:comment>Points to Authority class.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="PersonName">
    <rdfs:label>PersonName</rdfs:label>
    <rdfs:comment>Name of the person who is the authority cited.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="PositionTitle">
    <rdfs:label>PositionTitle</rdfs:label>
    <rdfs:comment>Position title of the person who is the authority
cited.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="OrganizationName">
    <rdfs:label>OrganizationName</rdfs:label>
    <rdfs:comment>Name of the organization of the person who is the authority
cited.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="OrganizationAddress">
    <rdfs:label>OrganizationAddress</rdfs:label>
    <rdfs:comment>Address of the organization of the person who is the
authority cited.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Date">
    <rdfs:label>Date</rdfs:label>
    <rdfs:comment>Date of the approval/signature.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="VersionName">
    <rdfs:label>VersionName</rdfs:label>
    <rdfs:comment>Name of the version.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Title">
    <rdfs:label>Title</rdfs:label>
    <rdfs:comment>Title of the KA/KE artifact or behavior.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Subject">
    <rdfs:label>Subject</rdfs:label>
    <rdfs:comment>Topic of the content, typically expressed as keywords, key
phrases, or classification codes.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Description">
    <rdfs:label>Description</rdfs:label>
    <rdfs:comment>An account of the content, I.e., a summary.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="ResourceIdentifier">
    <rdfs:label>ResourceIdentifier</rdfs:label>
    <rdfs:comment>Unambiguous reference to the resource within a given
context.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="SecurityClassification">
    <rdfs:label>SecurityClassification</rdfs:label>
```

```
     <rdfs:comment>Classification level of the KA/KE artifact or
behavior.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Releasability">
     <rdfs:label>Releasability</rdfs:label>
     <rdfs:comment>Description of who (individuals or organizations) may access
the KA/KE artifact or behavior.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasGeneralMetadata">
     <rdfs:label>hasGeneralMetadata</rdfs:label>
     <rdfs:comment>Points to a GeneralMetadata class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasParentDocument">
     <rdfs:label>hasParentDocument</rdfs:label>
     <rdfs:comment>Points to a KAKEArtifact class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCurrentVersion">
     <rdfs:label>hasCurrentVersion</rdfs:label>
     <rdfs:comment>Points to a Version class.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPriorVersion">
     <rdfs:label>hasPriorVersion</rdfs:label>
     <rdfs:comment>Points to a Version class.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="KEType">
     <rdfs:label>KEType</rdfs:label>
     <rdfs:comment>Boolean attribute for KAKEArtifact that is True if the
document is of the KE type only or combined KA/KE type only and False otherwise,
e.g., is of another type than strictly KE or strictly KA and KE
only.</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasKeyWord">
     <rdfs:label>hasKeyWord</rdfs:label>
     <rdfs:comment>Points to the KeyWord class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasKAKEType">
     <rdfs:label>hasKAKEType</rdfs:label>
     <rdfs:comment>Points to the KAKEType class.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
</rdf:RDF>
```

Attachment 5 - ConceptDomainMetadata Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
      xmlns:owl="http://www.w3.org/2002/07/owl#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
      xmlns:cdm="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl#"
      xml:base="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl#">
<owl:Ontology rdf:about="">
      <rdfs:comment>Describes the concept of the domain, including restrictions,
in which the behavior will operate.  Includes
ConceptDomainMetadata,AcceptableOrganizationalLevel,
AcceptableEntityType,Actor,Entity,Task,Unit, ConditionsDomain,Descriptor, and
ConditionsLevel</rdfs:comment>
      <owl:imports rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-
ont.owl"/>
      <rdfs:label>Concept Domain Metadata Ontology</rdfs:label>
</owl:Ontology>
<owl:Class rdf:ID="Actor">
     <rdfs:label>Actor</rdfs:label>
     <rdfs:comment>The entity responsible for performing the behavior. Contains
the information for the entity (Actor) performing the behavior plus links to the
Actor's unit and task.</rdfs:comment>
     <rdfs:subClassOf rdf:resource="#Entity"/>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty
rdf:resource="#MinimumNumberOfOtherEntitiesRequired"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ConceptDomainMetadata">
     <rdfs:label>ConceptDomainMetadata</rdfs:label>
     <rdfs:comment>Describes the concept of the domain in which the behavior
will operate.  Includes datatypes of Constraints (conditions for the behavior),
Goal (behavior objective), and Note (human readable text). It has object type
properties of hasAcceptableEntityType and hasAcceptableOrganizationalLevel for
entity types and organizational levels which can perform the behavior,
hasConditionsDomain for UJTL ConditionsDomain, hasInput and hasOutput for
inputs/outputs, hasActor for behavior performer, hasOtherEntity for other
involved entities, and hasMechanism for entities used by the
behavior.</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
```

```
                <owl:onProperty rdf:resource="#Constraints"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#Goal"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasActor"/>
                <owl:allValuesFrom rdf:resource="#Actor" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasConditionsDomain"/>
                <owl:allValuesFrom rdf:resource="#ConditionsDomain" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="#hasAcceptableOrganizationalLevel"/>
                <owl:allValuesFrom rdf:resource="#Unit" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasAcceptableEntityType"/>
                <owl:allValuesFrom rdf:resource="#Entity" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasAcceptableSide"/>
                <owl:allValuesFrom rdf:resource="#Side" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#hasInput"
/>
                <owl:allValuesFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#Input" />
```

```
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#hasOutput"
/>
               <owl:allValuesFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#Output" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#hasMechanism"/>
               <owl:allValuesFrom rdf:resource="#Entity" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#hasOtherEntity"/>
               <owl:allValuesFrom rdf:resource="#Entity" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ConditionsDomain">
     <rdfs:label>ConditionsDomain</rdfs:label>
     <rdfs:comment>The description of the enironmental conditions in the
simulation within which the behavior is executed that must be, may be or cannot
be present for the behavior to be validly executed.  Has data type properties of
Description and Note and object type properties of
hasCurrentLevel(ConditionsLevel) and hasRequiredDescriptor(Descriptor),
hasDescriptor(Descriptor), and
hadProhibitedDescriptor(Descriptor).</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#Note"/>
               <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty
rdf:resource="#ConditionsDomainLevelDescription"/>
               <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#hasRequiredDescriptor"/>
               <owl:allValuesFrom rdf:resource="#Descriptor" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
               <owl:onProperty rdf:resource="#hasProhibitedDescriptor"/>
```

```
                <owl:allValuesFrom rdf:resource="#Descriptor" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCurrentLevel"/>
                <owl:allValuesFrom rdf:resource="#ConditionsLevel" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasDescriptor"/>
                <owl:allValuesFrom rdf:resource="#Descriptor" />
            </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ConditionsLevel">
        <rdfs:label>ConditionsLevel</rdfs:label>
        <rdfs:comment>Contains information for the current ConditionsDomain
restriction and upper levels of the ConditionsDomain.  Information includes data
type properties of IsTopLevel, the LevelTitle, LevelIDNumber, and a Note plus an
object type property of hasNextUpperLevel(ConditionsLevel).</rdfs:comment>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#IsTopLevel"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#boolean" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#LevelTitle"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#LevelIDNumber"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasNextUpperLevel"/>
                <owl:allValuesFrom rdf:resource="#ConditionsLevel" />
            </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:ID="Descriptor">
     <rdfs:label>Descriptor</rdfs:label>
     <rdfs:comment>The class that contains a list of descriptors from the UJTL
that may be either permitted, prohibited or required.  It has data type
properties of Name, Description and Note.</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#DescriptorName"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#DescriptorDescription"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Entity">
     <rdfs:label>Entity</rdfs:label>
     <rdfs:comment>An object in the simulation that could be a human, a vehicle,
a structure, a weapon, of anything else which can exist, affect the actions of
the simulation, and be specifically identified.  It includes datatypes for
Entity ID, EntityType, MinimumNumberRequired of that type of entity required for
the behavior's concept domain, EntityStatus for the status of the entity and a
Note plus object type properties for hasTask(Task) and belongsToUnit(Unit).  It
also has a subclass of Actor.</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#EntityID"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#EntityStatus"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
```

```
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#EntityType"/>
            <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasSide"/>
            <owl:allValuesFrom rdf:resource="#Side" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#MinimumNumberRequired"/>
            <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#belongsToUnit" />
            <owl:allValuesFrom rdf:resource="#Unit" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasTask" />
            <owl:allValuesFrom rdf:resource="#Task" />
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Side">
    <rdfs:label>Side</rdfs:label>
    <rdfs:comment>A class that describes a side available to a Unit or
Entity</rdfs:comment>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasFriendSide"/>
            <owl:allValuesFrom rdf:resource="#Side" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasOpponentSide"/>
            <owl:allValuesFrom rdf:resource="#Side" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasNeutralSide"/>
            <owl:allValuesFrom rdf:resource="#Side" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
```

```
                <owl:onProperty rdf:resource="#SideName"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Task">
     <rdfs:label>Task</rdfs:label>
     <rdfs:comment>A low level operation to be accomplished in whole or in part
by the behavior of the Actor and the other human/unit entities using "mechanism"
entities.  Entities and Units have associated tasks.  It has data type
properties of TaskID and Note, plus an object type property of
performsUsingEntity(Entity).</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#TaskID"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#performsUsingEntity" />
                <owl:allValuesFrom rdf:resource="#Entity" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Unit">
     <rdfs:label>Unit</rdfs:label>
     <rdfs:comment>An organization composed of individuals and/or other units
and which may also be a part of a higher level unit and have subordinate units.
Units also have associated tasks.  It has data type properties of UnitID,
UnitLevelType and Note, plus object type properties of hasTask(Task),
hasUnitEntity(Entity), hasHigherUnit(Unit), and
hasSubordinateUnit(Unit).</rdfs:comment>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#UnitID"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
```

```
          <owl:Restriction>
                <owl:onProperty rdf:resource="#UnitLevelType"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#Note"/>
                <owl:allValuesFrom
rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasSide"/>
                <owl:allValuesFrom rdf:resource="#Side" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasTask" />
                <owl:allValuesFrom rdf:resource="#Task" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnitEntity" />
                <owl:allValuesFrom rdf:resource="#Entity" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasHigherUnit" />
                <owl:allValuesFrom rdf:resource="#Unit" />
          </owl:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
          <owl:Restriction>
                <owl:onProperty rdf:resource="#hasSubordinateUnit" />
                <owl:allValuesFrom rdf:resource="#Unit" />
          </owl:Restriction>
     </rdfs:subClassOf>
</owl:Class>
<owl:DatatypeProperty rdf:ID="Constraints">
     <rdfs:label>Constraints</rdfs:label>
     <rdfs:comment>A list of conditions with which the behavior must
commply.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Goal">
     <rdfs:label>Goal</rdfs:label>
     <rdfs:comment>Objective that the behavior is executing to
achieve.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Note">
     <rdfs:label>Note</rdfs:label>
     <rdfs:comment>A human readable comment.</rdfs:comment>
```

```
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasConceptDomainMetadata">
     <rdfs:label>hasConceptDomainMetadata</rdfs:label>
     <rdfs:comment>Points to ConceptDomainMetadata class.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="MinimumNumberOfOtherEntitiesRequired">
     <rdfs:label>MinimumNumberOfOtherEntitiesRequired</rdfs:label>
     <rdfs:comment>Minimum number of entities, other than the Actor, required in
the behavior.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="EntityID">
     <rdfs:label>EntityID</rdfs:label>
     <rdfs:comment>ID for the entity.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="EntityType">
     <rdfs:label>EntityType</rdfs:label>
     <rdfs:comment>Entity type of the entity.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="EntityStatus">
     <rdfs:label>EntityStatus</rdfs:label>
     <rdfs:comment>Status of the entity, e.g., normal, destroyed,
etc.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="UnitID">
     <rdfs:label>UnitID</rdfs:label>
     <rdfs:comment>ID for the unit.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="UnitLevelType">
     <rdfs:label>UnitLevelType</rdfs:label>
     <rdfs:comment>Type of the Unit Level, e.g., individual, platoon, squadron,
division, etc.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="TaskID">
     <rdfs:label>TaskID</rdfs:label>
     <rdfs:comment>ID for the task, e.g., from the Universal Joint Task
List</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="MinimumNumberRequired">
     <rdfs:label>MinimumNumberRequired</rdfs:label>
     <rdfs:comment>Minimum number of entities of the same type
required.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasActor">
     <rdfs:label>hasActor</rdfs:label>
     <rdfs:comment>Points to Actor class.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasMechanism">
     <rdfs:label>hasMechanism</rdfs:label>
```

```
    <rdfs:comment>Points to Entity class for a mechanism used in the
behavior.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasOtherEntity">
    <rdfs:label>hasOtherEntity</rdfs:label>
    <rdfs:comment>Points to the Entity class for an entity, other than the
Actor, which is involved in the behavior.  There may be multiple other
entities.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasUnitEntity">
    <rdfs:label>hasUnitEntity</rdfs:label>
    <rdfs:comment>Points to an Entity from the Unit</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasTask">
    <rdfs:label>hasTask</rdfs:label>
    <rdfs:comment>Points to Task class for entities and units.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasHigherUnit">
    <rdfs:label>hasHigherUnit</rdfs:label>
    <rdfs:comment>Points to unit that is above the unit referencing
it.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSubordinateUnit">
    <rdfs:label>hasSubordinateUnit</rdfs:label>
    <rdfs:comment>Points to unit that is subordinate to the unit referencing
it.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="performsUsingEntity">
    <rdfs:label>performsUsingEntity</rdfs:label>
    <rdfs:comment>Points to entity used by the Task.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="belongsToUnit">
    <rdfs:label>belongsToUnit</rdfs:label>
    <rdfs:comment>Points to the Unit assigned to the entity.</rdfs:comment>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="ConditionsDomainLevelDescription">
    <rdfs:label>ConditionsDomainLevelDescription</rdfs:label>
    <rdfs:comment>Descriiption of a specific UJTL Conditions Domain
level.</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="DescriptorDescription">
    <rdfs:label>DescriptorDescription</rdfs:label>
    <rdfs:comment>Description of a specific UJTL Descriptor from the given
ConditionsDomain level.</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="IsTopLevel">
    <rdfs:label>IsTopLevel</rdfs:label>
    <rdfs:comment>Boolean description of whether or not the ConditionsLevel is
the top level.</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="LevelTitle">
    <rdfs:label>LevelTitle</rdfs:label>
    <rdfs:comment>Name of a specific UJTL Conditions Domain
level.</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
```

```
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="LevelIDNumber">
     <rdfs:label>LevelIDNumber</rdfs:label>
     <rdfs:comment>String displaying the UJTL conditions level ID number, a
series of integers separated by decimals.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasAcceptableEntityType">
     <rdfs:label>hasAcceptableEntityType</rdfs:label>
     <rdfs:comment>Points to an Entity class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasConditionsDomain">
     <rdfs:label>hasConditionsDomain</rdfs:label>
     <rdfs:comment>Points to a ConditionsDomain class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasAcceptableOrganizationalLevel">
     <rdfs:label>hasAcceptableOrganizationalLevel</rdfs:label>
     <rdfs:comment>Points to a Unit class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasRequiredDescriptor">
     <rdfs:label>hasRequiredDescriptor</rdfs:label>
     <rdfs:comment>Points to the Descriptor class for required UJTL Conditions
descriptors.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasProhibitedDescriptor">
     <rdfs:label>hasProhibitedDescriptor</rdfs:label>
     <rdfs:comment>Points to the Descriptor class for prohibited UJTL Conditions
descriptors.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasCurrentLevel">
     <rdfs:label>hasCurrentLevel</rdfs:label>
     <rdfs:comment>Points to the ConditionsLevel class.</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasNextUpperLevel">
     <rdfs:label>hasNextUpperLevel</rdfs:label>
     <rdfs:comment>Points to the next upper level Conditions Domain
level.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasDescriptor">
     <rdfs:label>hasDescriptor</rdfs:label>
     <rdfs:comment>Points to the Descriptor class for descriptors from the UJTL
Conditions descriptors that are neither required nor prohibited for the
Behavior..</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="DescriptorName">
     <rdfs:label>DescriptorName</rdfs:label>
     <rdfs:comment>Name of one of the descriptors of the Conditions Domain for
the level it's on.</rdfs:comment>
     <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="SideName">
     <rdfs:label>SideName</rdfs:label>
     <rdfs:comment>Name of the Side</rdfs:comment>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasSide">
     <rdfs:label>hasSide</rdfs:label>
```

```
     <rdfs:comment>Points to a Side class</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasFriendSide">
     <rdfs:label>hasFriendSide</rdfs:label>
     <rdfs:comment>Points to a Side class</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasOpponentSide">
     <rdfs:label>hasOpponentSide</rdfs:label>
     <rdfs:comment>Points to a Side class</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasNeutralSide">
     <rdfs:label>hasNeutralSide</rdfs:label>
     <rdfs:comment>Points to a Side class</rdfs:comment>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasAcceptableSide">
     <rdfs:label>hasAcceptableSide</rdfs:label>
     <rdfs:comment>Points to a Side class</rdfs:comment>
</owl:ObjectProperty>
</rdf:RDF>
```

Attachment 6–main.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:var="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns:cdm="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <!-- External XSLTs that contain templates called/applied within this XSLT
-->
      <xsl:import href="primitive.xslt"/>
      <xsl:import href="composite.xslt"/>
      <xsl:import href="artifact.xslt"/>
      <xsl:import href="conceptDomainMetadata.xslt"/>

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <!-- XSLTs importing main.xslt will make use of $extURI. -->
      <xsl:variable
name="extURI">http://orlando.drc.com/hbr/artifacts/</xsl:variable>

      <!-- Entry Point -->
      <xsl:template match="/">
            <xsl:call-template name="rdf-root-node"/>
      </xsl:template>

      <!-- Create the root rdf:RDF node and attributes -->
      <xsl:template name="rdf-root-node">
            <rdf:RDF>
                  <!--External Template - Template found in primitive.xslt-->
                  <xsl:apply-templates
select="net.onesaf.services.model.representation.behavior.PrimitiveBehaviorDescr
iption"/>
                  <!--External Template - Template found in composite.xslt-->
                  <xsl:apply-templates
select="net.onesaf.services.model.representation.behavior.CompositeBehaviorDescr
iption"/>
                  <!--External Template - Template found in artifact.xslt-->
                  <xsl:call-template name="artifact"/>
                  <!--External Template - Template found in
conceptDomainMetadata.xslt-->
                  <xsl:if test="count(//metaData/*/*/string) &gt; 0">
                        <xsl:call-template name="conceptDomainMetadata"/>
                  </xsl:if>
            </rdf:RDF>
      </xsl:template><!--End of template name="rdf-root-node"-->

</xsl:stylesheet>
```

Attachment 7 - artifact.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#">

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <xsl:template name="artifact">
            <art:GeneralMetadata rdf:ID="GenMetadata-0">
                  <art:Title/>
                  <art:Subject/>
                  <art:SecurityClassification>Official Use
Only</art:SecurityClassification>
                  <art:Releasability>U.S. Government
Contractors</art:Releasability>
                  <art:Description>
                        <xsl:value-of select="//semanticDescription"/>
                  </art:Description>
            </art:GeneralMetadata>
            <art:Version rdf:ID="Ver-0">
                  <art:VersionDate>
                        <xsl:value-of select="//validationDate"/>
                  </art:VersionDate>
                  <art:hasVVARecord rdf:resource="#VVARec-0"/>
            </art:Version>
            <art:VVARecord rdf:ID="VVARec-0">
                  <art:VerificationComplete/>
                  <art:ValidationComplete>
                        <xsl:variable name="validationState"
select="//validationState"/>
                        <xsl:choose>
                              <xsl:when
test="$validationState='VALIDATED'">True</xsl:when>
                              <xsl:otherwise>False</xsl:otherwise>
                        </xsl:choose>
                  </art:ValidationComplete>
                  <art:AccredidationComplete/>
                  <art:hasVerificationAuthority rdf:resource="#Auth-0"/>
                  <art:hasValidationAuthority rdf:resource="#Auth-0"/>
                  <art:hasAccredidationAuthority rdf:resource="#Auth-0"/>
            </art:VVARecord>
            <art:Authority rdf:ID="Auth-0">
                  <art:PersonName/>
                  <art:PositionTitle/>
                  <art:OrganizationName/>
            </art:Authority>
      </xsl:template>

</xsl:stylesheet>
```

Attachment 8 - composite.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns:cdm="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <!-- External XSLTs that contain templates called/applied within this XSLT
-->
      <xsl:import href="variable.xslt"/>
      <xsl:import href="timeline-container.xslt"/>

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <!-- Composite Behavior Structure -->
      <xsl:template
match="net.onesaf.services.model.representation.behavior.CompositeBehaviorDescri
ption">
            <xsl:variable name="behaviorName" select="//name"/>

            <beh:CompositeBehavior>
                  <xsl:attribute name="rdf:ID">
                        <xsl:value-of
select="translate($behaviorName,'/','.')"/>
                  </xsl:attribute>

                  <!-- These elements are hard coded until there is information
available. -->
                  <beh:DraftBehavior>True</beh:DraftBehavior>
                  <beh:Resolution/>

                  <xsl:choose>
                        <xsl:when
test="contains($behaviorName,'behavior/composite/lf')">
                              <beh:Fidelity>Low</beh:Fidelity>
                        </xsl:when>
                        <xsl:when
test="contains($behaviorName,'behavior/composite/hf')">
                              <beh:Fidelity>High</beh:Fidelity>
                        </xsl:when  >
                  </xsl:choose>

                  <!-- All the oos composite behaviors have a
TemporalRelationElement as the child to executionTimeline. -->
                  <!-- Therefore, the object property hasTopLevelContainer will
always be point a container instance. -->
                  <beh:hasTopLevelContainer>
                        <xsl:attribute name="rdf:resource">
                              <xsl:value-of select="concat('#Container-
',executionTimeline/*[1]/@refID)"/>
                        </xsl:attribute>
                  </beh:hasTopLevelContainer>
```

```
                <!-- The art: references are hard coded until there is general
metadata and version information available. -->
                <art:hasGeneralMetadata rdf:resource="#GenMetadata-0"/>
                <art:hasCurrentVersion rdf:resource="#Ver-0"/>

                <!-- The cdm: references come from the metadata element for
unit information -->
                <!-- Call only if data within string element exists. -->
                <xsl:if test="count(//metaData/*/*/string) &gt; 0">
                      <cdm:hasConceptDomainMetadata
rdf:resouce="#ConceptDomainMetadata-0"/>
                </xsl:if>

                <!-- Defining objectproperties which reference others classes
to be instantiated by another template -->
                <!--External Template - Template found in variable.xslt-->
                <xsl:apply-templates
select="table/*/entries/net.onesaf.services.model.representation.behavior.TableE
ntry" mode="reference"/>
            </beh:CompositeBehavior>

            <!--External Template - Template found in timeline.xslt-->
            <xsl:apply-templates
select="executionTimeline/net.onesaf.services.model.representation.behavior.Temp
oralRelationElement" mode="instance">
                <xsl:with-param name="toplevel">true</xsl:with-param>
            </xsl:apply-templates>

            <!--External Template - Template found in variable.xslt-->
            <xsl:apply-templates
select="table/*/entries/net.onesaf.services.model.representation.behavior.TableE
ntry" mode="instance"/>

      </xsl:template>

</xsl:stylesheet>
```

Attachment 9 - conceptDomainMetadata.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:cdm="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl#">

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <xsl:template name="conceptDomainMetadata">
            <cdm:ConceptDomainMetadata rdf:ID="ConceptDomainMetadata-0">
                  <xsl:apply-templates select="//metaData/*/doctrineList"
mode="Reference"/>
                  <xsl:apply-templates select="//metaData/*/echelonList"
mode="Reference"/>
                  <xsl:apply-templates select="//metaData/*/unitTypeList"
mode="Reference"/>
            </cdm:ConceptDomainMetadata>
            <xsl:apply-templates select="//metaData/*/doctrineList"
mode="Instance"/>
            <xsl:apply-templates select="//metaData/*/echelonList"
mode="Instance"/>
            <xsl:apply-templates select="//metaData/*/unitTypeList"
mode="Instance"/>
      </xsl:template>

      <xsl:template match="doctrineList" mode="Reference">
            <xsl:if test="count(*) &gt; 0">
                  <xsl:for-each select="string">
                        <cdm:hasAcceptableSide>
                              <xsl:attribute name="rdf:resouce">
                                    <xsl:value-of select="concat('#Side-
',position())"/>
                              </xsl:attribute>
                        </cdm:hasAcceptableSide>
                  </xsl:for-each>
            </xsl:if>
      </xsl:template>
      <xsl:template match="doctrineList" mode="Instance">
            <xsl:if test="count(*) &gt; 0">
                  <xsl:for-each select="string">
                        <cdm:Side>
                              <xsl:attribute name="rdf:ID">
                                    <xsl:value-of select="concat('Side-
',position())"/>
                              </xsl:attribute>
                              <cdm:SideName>
                                    <xsl:value-of select="."/>
                              </cdm:SideName>
                        </cdm:Side>
                  </xsl:for-each>
            </xsl:if>
      </xsl:template>

      <xsl:template match="echelonList" mode="Reference">
            <xsl:if test="count(*) &gt; 0">
                  <xsl:for-each select="string">
```

```
                        <cdm:hasAcceptableOrganizationalLevel>
                              <xsl:attribute name="rdf:resouce">
                                    <xsl:value-of select="concat('#Unit-
',position())"/>
                              </xsl:attribute>
                        </cdm:hasAcceptableOrganizationalLevel>
                  </xsl:for-each>
            </xsl:if>
      </xsl:template>
      <xsl:template match="echelonList" mode="Instance">
            <xsl:if test="count(*) &gt; 0">
                  <xsl:for-each select="string">
                        <cdm:Unit>
                              <xsl:attribute name="rdf:ID">
                                    <xsl:value-of select="concat('Unit-
',position())"/>
                              </xsl:attribute>
                              <cdm:UnitName>
                                    <xsl:value-of select="."/>
                              </cdm:UnitName>
                        </cdm:Unit>
                  </xsl:for-each>
            </xsl:if>
      </xsl:template>

      <xsl:template match="unitTypeList" mode="Instance">
            <xsl:if test="count(*) &gt; 0">
                  <xsl:for-each select="string">
                        <cdm:Entity>
                              <xsl:attribute name="rdf:ID">
                                    <xsl:value-of select="concat('Entity-
',position())"/>
                              </xsl:attribute>
                              <cdm:EntityType>
                                    <xsl:value-of select="."/>
                              </cdm:EntityType>
                        </cdm:Entity>
                  </xsl:for-each>
            </xsl:if>
      </xsl:template>
      <xsl:template match="unitTypeList" mode="Reference">
            <xsl:if test="count(*) &gt; 0">
                  <xsl:for-each select="string">
                        <cdm:hasAcceptableEntityType>
                              <xsl:attribute name="rdf:resouce">
                                    <xsl:value-of select="concat('#Entity-
',position())"/>
                              </xsl:attribute>
                        </cdm:hasAcceptableEntityType>
                  </xsl:for-each>
            </xsl:if>
      </xsl:template>
</xsl:stylesheet>
```

Attachment 10 - primitive.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <!-- External XSLT that contain templates called/applied within this XSLT
-->
      <xsl:import href="variable.xslt"/>

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <!-- Primative Behavior Structure -->
      <xsl:template
match="net.onesaf.services.model.representation.behavior.PrimitiveBehaviorDescri
ption">
            <beh:PrimitiveBehavior>
                  <xsl:attribute name="rdf:ID">
                        <xsl:value-of
select="translate(/net.onesaf.services.model.representation.behavior.PrimitiveBe
haviorDescription/name,'/','.')"/>
                  </xsl:attribute>
                  <beh:SoftwareLanguage>Java</beh:SoftwareLanguage>
                  <beh:DraftBehavior>True</beh:DraftBehavior>
                  <beh:Resolution/>
                  <xsl:variable name="behaviorName" select="//name"/>
                  <xsl:choose>
                        <xsl:when
test="contains($behaviorName,'behavior/primitive/lf')">
                              <beh:Fidelity>Low</beh:Fidelity>
                        </xsl:when>
                        <xsl:otherwise>
                              <xsl:if
test="contains($behaviorName,'behavior/primitive/hf')">
                                    <beh:Fidelity>High</beh:Fidelity>
                              </xsl:if>
                        </xsl:otherwise>
                  </xsl:choose>

                  <!-- The art: references are hard coded until there is general
metadata and version information available. -->
                  <art:hasGeneralMetadata rdf:resource="#GenMetadata-0"/>
                  <art:hasCurrentVersion rdf:resource="#Ver-0"/>

                  <!-- Defining objectproperties which reference others classes
to be instantiated by another template -->
                  <!--External Template - Template found in variable.xslt-->
                  <xsl:apply-templates
select="table/net.onesaf.services.model.representation.behavior.TableEntries/ent
ries/net.onesaf.services.model.representation.behavior.TableEntry"
mode="reference"/>
            </beh:PrimitiveBehavior>

            <!--External Template - Template found in variable.xslt-->
```

```
            <xsl:apply-templates
select="table/net.onesaf.services.model.representation.behavior.TableEntries/ent
ries/net.onesaf.services.model.representation.behavior.TableEntry"
mode="instance"/>

      </xsl:template>

</xsl:stylesheet>
```

Attachment 11 - timeline-common.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <xsl:template name="internalBehaviorName">
            <xsl:param name="behaviorType"/>
            <xsl:param name="behaviorName"/>
            <xsl:choose>
                  <xsl:when test="contains($behaviorName,'/lf/')">
                        <xsl:choose>
                              <xsl:when test="contains($behaviorName,'.xml')">
                                    <xsl:value-of select="concat(substring-
after(substring-
before($behaviorName,'.xml'),concat('behavior/',$behaviorType,'/lf/')),'-
',@refID)"/>
                              </xsl:when>
                              <xsl:otherwise>
                                    <xsl:value-of select="concat(substring-
after($behaviorName,concat('behavior/',$behaviorType,'/lf/')),'-',@refID)"/>
                              </xsl:otherwise>
                        </xsl:choose>
                  </xsl:when>
                  <xsl:when test="contains($behaviorName,'/hf/')">
                        <xsl:choose>
                              <xsl:when test="contains($behaviorName,'.xml')">
                                    <xsl:value-of select="concat(substring-
after(substring-
before($behaviorName,'.xml'),concat('behavior/',$behaviorType,'/hf/')),'-
',@refID)"/>
                              </xsl:when>
                              <xsl:otherwise>
                                    <xsl:value-of select="concat(substring-
after($behaviorName,concat('behavior/',$behaviorType,'/hf/')),'-',@refID)"/>
                              </xsl:otherwise>
                        </xsl:choose>
                  </xsl:when>
            </xsl:choose>
      </xsl:template>

      <xsl:template name="internalOrderSenderName">
            <xsl:param name="behaviorType"/>
            <xsl:param name="behaviorName"/>
            <xsl:choose>
                  <xsl:when test="contains($behaviorName,'/lf/')">
                        <xsl:value-of select="concat(substring-
after($behaviorName,concat('behavior/',$behaviorType,'/lf/')),'-OS-',@refID)"/>
                  </xsl:when>
                  <xsl:when test="contains($behaviorName,'/hf/')">
                        <xsl:value-of select="concat(substring-
after($behaviorName,concat('behavior/',$behaviorType,'/hf/')),'-OS-',@refID)"/>
                  </xsl:when>
```

```
            </xsl:choose>
      </xsl:template>

      <!-- Currently called by primitive instance - need to incorporate the
others (but that may lead to another template -->
      <xsl:template name="externalBehaviorName">
            <xsl:attribute name="rdf:resource">
                  <xsl:variable name="behaviorName" select="behaviorName"/>
                  <!-- Use of $extURI which is defined in main.xslt -->
                  <xsl:value-of select="concat($extURI,substring-
after($behaviorName,'behavior/'),'.rdf#',translate($behaviorName,'/','.'))"/>
            </xsl:attribute>
      </xsl:template>

</xsl:stylesheet>
```

Attachment 12 - timeline-compositeBehavior.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <!-- External XSLT that contain templates called/applied within this XSLT
-->
      <xsl:import href="variable.xslt"/>
      <xsl:import href="timeline-common.xslt"/>

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.CompositeBehaviorElemen
t" mode="firstComponent">
            <beh:hasFirstComponent>
                  <xsl:call-template name="defineAttributeForCBeh"/>
            </beh:hasFirstComponent>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.CompositeBehaviorElemen
t" mode="reference">
            <beh:hasCompositeBehavior>
                  <xsl:call-template name="defineAttributeForCBeh"/>
            </beh:hasCompositeBehavior>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.CompositeBehaviorElemen
t" mode="nextComponent">
            <beh:hasNextComponent>
                  <xsl:call-template name="defineAttributeForCBeh"/>
            </beh:hasNextComponent>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.CompositeBehaviorElemen
t" mode="truePath">
            <beh:hasTrueBranchPath>
                  <xsl:call-template name="defineAttributeForCBeh"/>
            </beh:hasTrueBranchPath>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.CompositeBehaviorElemen
t" mode="falsePath">
            <beh:hasFalseBranchPath>
                  <xsl:call-template name="defineAttributeForCBeh"/>
            </beh:hasFalseBranchPath>
      </xsl:template>

      <xsl:template name="defineAttributeForCBeh">
            <xsl:attribute name="rdf:resource">
```

```
                  <xsl:value-of select="string('#')"/>
                  <!--External Template - Template found in timeline-
common.xslt-->
                  <xsl:call-template name="internalBehaviorName">
                        <xsl:with-param name="behaviorType"
select="string('composite')"/>
                        <xsl:with-param name="behaviorName"
select="behaviorName"/>
                  </xsl:call-template>
            </xsl:attribute>
      </xsl:template>

      <xsl:template name="CompositeBehaviorElementInst">
            <xsl:variable name="behaviorName" select="behaviorName"/>

            <xsl:attribute name="rdf:ID">
                  <!--External Template - Template found in timeline-
common.xslt-->
                  <xsl:call-template name="internalBehaviorName">
                        <xsl:with-param name="behaviorType"
select="string('composite')"/>
                        <xsl:with-param name="behaviorName"
select="$behaviorName"/>
                  </xsl:call-template>
            </xsl:attribute>

            <beh:usesBehaviorOf>
                  <!--External Template - Template found in timeline-
common.xslt-->
                  <xsl:call-template name="externalBehaviorName"/>
            </beh:usesBehaviorOf>
      </xsl:template>

</xsl:stylesheet>
```

Attachment 13 - timeline-conditionalBranch.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <!-- External XSLT that contain templates called/applied within this XSLT
-->
      <xsl:import href="variable.xslt"/>
      <xsl:import href="timeline-common.xslt"/>

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <xsl:template
      match="net.onesaf.services.model.representation.behavior.ConditionalBranch
Element" mode="firstComponent">
            <beh:hasFirstComponent>
                  <xsl:call-template name="defineAttributeForCBranch"/>
            </beh:hasFirstComponent>
      </xsl:template>

      <xsl:template
      match="net.onesaf.services.model.representation.behavior.ConditionalBranch
Element" mode="reference">
            <beh:hasConditionalBranch>
                  <xsl:call-template name="defineAttributeForCBranch"/>
            </beh:hasConditionalBranch>
            <!--External Template - Template called depends on first child of
path. -->
            <xsl:apply-templates select="falsePath/*[1]" mode="reference"/>
            <xsl:apply-templates select="truePath/*[1]" mode="reference"/>
      </xsl:template>

      <xsl:template
      match="net.onesaf.services.model.representation.behavior.ConditionalBranch
Element" mode="nextComponent">
            <beh:hasNextComponent>
                  <xsl:call-template name="defineAttributeForCBranch"/>
            </beh:hasNextComponent>
      </xsl:template>

      <xsl:template
      match="net.onesaf.services.model.representation.behavior.ConditionalBranch
Element" mode="truePath">
            <beh:hasTrueBranchPath>
                  <xsl:call-template name="defineAttributeForCBranch"/>
            </beh:hasTrueBranchPath>
      </xsl:template>

      <xsl:template
      match="net.onesaf.services.model.representation.behavior.ConditionalBranch
Element" mode="falsePath">
            <beh:hasFalseBranchPath>
                  <xsl:call-template name="defineAttributeForCBranch"/>
            </beh:hasFalseBranchPath>
```

```
    </xsl:template>

    <xsl:template name="defineAttributeForCBranch">
        <xsl:attribute name="rdf:resource">
            <xsl:value-of select="concat('#ConditionalBranch-',@refID)"/>
        </xsl:attribute>
    </xsl:template>

    <xsl:template name="CB-ConditionalExpressionRef">
        <xsl:attribute name="rdf:ID">
            <xsl:value-of select="concat('ConditionalBranch-',@refID)"/>
        </xsl:attribute>
        <beh:hasConditionalExpression>
            <xsl:attribute name="rdf:resource">
                <xsl:value-of select="concat('#ConditionalExpression-
',*[1]/*[1]/@refID)"/>
            </xsl:attribute>
        </beh:hasConditionalExpression>
    </xsl:template>

    <xsl:template
match="net.onesaf.services.model.representation.behavior.AtomicPredicateElement"
mode="instance">
        <xsl:param name="postCondLoop"/>
        <beh:ConditionalExpression>
            <xsl:attribute name="rdf:ID">
                <xsl:value-of select="concat('ConditionalExpression-
',@refID)"/>
            </xsl:attribute>

            <beh:DefaultValue/>
            <beh:hasConditionalPredicate>
                <xsl:attribute name="rdf:resource">
                    <xsl:value-of select="string('#')"/>
                    <!--External Template - Template found in
timeline-common.xslt-->
                    <xsl:call-template name="internalBehaviorName">
                        <xsl:with-param name="behaviorType"
select="string('primitive')"/>
                        <xsl:with-param name="behaviorName"
select="predicateDescriptionName"/>
                    </xsl:call-template>
                </xsl:attribute>
            </beh:hasConditionalPredicate>
        </beh:ConditionalExpression>

        <!--External Template - Template found in timeline-primitive.xslt-->
        <xsl:call-template name="ConditionPrimitiveInstance">
            <xsl:with-param name="behaviorName"
select="predicateDescriptionName"/>
            <xsl:with-param name="postCondLoop" select="$postCondLoop"/>
        </xsl:call-template>
    </xsl:template>

</xsl:stylesheet>
```

Attachment 14 - timeline-container.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <!-- External XSLT that contain templates called/applied within this XSLT
-->
      <xsl:import href="variable.xslt"/>
      <xsl:import href="timeline-primitiveBehavior.xslt"/>
      <xsl:import href="timeline-conditionalBranch.xslt"/>
      <xsl:import href="timeline-orderSender.xslt"/>
      <xsl:import href="timeline-compositeBehavior.xslt"/>
      <xsl:import href="timeline-postConditionalLoop.xslt"/>

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.TemporalRelationElement
" mode="firstComponent">
            <beh:hasFirstComponent>
                  <xsl:call-template name="defineAttributeForCont"/>
            </beh:hasFirstComponent>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.TemporalRelationElement
" mode="reference">
            <beh:hasContainer>
                  <xsl:call-template name="defineAttributeForCont"/>
            </beh:hasContainer>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.TemporalRelationElement
" mode="nextComponent">
            <beh:hasNextComponent>
                  <xsl:call-template name="defineAttributeForCont"/>
            </beh:hasNextComponent>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.TemporalRelationElement
" mode="truePath">
            <beh:hasTrueBranchPath>
                  <xsl:call-template name="defineAttributeForCont"/>
            </beh:hasTrueBranchPath>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.TemporalRelationElement
" mode="falsePath">
            <beh:hasFalseBranchPath>
                  <xsl:call-template name="defineAttributeForCont"/>
            </beh:hasFalseBranchPath>
```

```
	</xsl:template>

	<xsl:template name="defineAttributeForCont">
		<xsl:attribute name="rdf:resource">
			<xsl:value-of select="concat('#Container-',@refID)"/>
		</xsl:attribute>
	</xsl:template>

	<xsl:template match="timelineElements" mode="reference">
		<!-- The for-each is necesary to preserve the order of the
sequential timeline. -->
		<xsl:for-each select="child::*">
			<!--External Template - Template found in timeline-
primitiveBehavior.xslt-->
			<xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.PrimitiveBehavio
rElement" mode="reference"/>
			<!--External Template - Template found in timeline-
conditionalBranch.xslt-->
			<xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.ConditionalBranc
hElement" mode="reference"/>
			<xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.TemporalRelation
Element" mode="reference"/>
			<!--External Template - Template found in timeline-
orderSender.xslt-->
			<xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.OrderSenderEleme
nt" mode="reference"/>
			<!--External Template - Template found in timeline-
compositeBehavior.xslt-->
			<xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.CompositeBehavio
rElement" mode="reference"/>
			<!--External Template - Template found in timeline-
postConditionalLoop.xslt-->
			<xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.PostConditionalL
oopElement" mode="reference"/>
		</xsl:for-each>
	</xsl:template>

	<xsl:template match="timelineElements" mode="instance">
		<!-- The for-each is necesary to preserve the order of the
sequential timeline. -->
		<xsl:for-each select="child::*">
			<xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.PrimitiveBehavio
rElement" mode="instance"/>
			<xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.ConditionalBranc
hElement" mode="instance"/>
			<xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.TemporalRelation
Element" mode="instance"/>
```

```
                  <xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.OrderSenderEleme
nt" mode="instance"/>
                  <xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.CompositeBehavio
rElement" mode="instance"/>
                  <xsl:apply-templates
select="self::net.onesaf.services.model.representation.behavior.PostConditionalL
oopElement" mode="instance"/>
            </xsl:for-each>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.TemporalRelationElement
" mode="instance">
            <xsl:param name="toplevel"/>
            <xsl:param name="branchPath"/>
            <xsl:param name="postCondLoop"/>

            <beh:Container>
                  <xsl:attribute name="rdf:ID">
                        <xsl:value-of select="concat('Container-',@refID)"/>
                  </xsl:attribute>
                  <beh:ContainerTitle>
                        <xsl:value-of select="elementIdentifier"/>
                  </beh:ContainerTitle>
                  <beh:ParallelContainerType>
                        <xsl:choose>
                              <xsl:when test="temporalType='SEQUENTIAL'">
                                    <xsl:text>False</xsl:text>
                              </xsl:when>
                              <xsl:otherwise>
                                    <xsl:text>True</xsl:text>
                              </xsl:otherwise>
                        </xsl:choose>
                  </beh:ParallelContainerType>

                  <!-- These elements are hard coded until there is information
 available. -->
                  <beh:InBackground>False</beh:InBackground>
                  <beh:Note/>

                  <!-- Only instantiate TableEntry references for the top level
 container. -->
                  <xsl:if test="$toplevel='true'">
                        <!--External Template - Template found in variable.xslt-
->
                        <xsl:apply-templates
select="//net.onesaf.services.model.representation.behavior.TableEntry"
mode="containerRefVars"/>
                  </xsl:if>

                  <xsl:apply-templates select="timelineElements/*[1]"
mode="firstComponent"/>
                  <xsl:apply-templates select="timelineElements"
mode="reference"/>
```

```
                  <xsl:call-template name="nextComponent">
                        <xsl:with-param name="branchPath" select="$branchPath"/>
                        <xsl:with-param name="postCondLoop"
select="$postCondLoop"/>
                  </xsl:call-template>

            </beh:Container>

            <xsl:apply-templates select="timelineElements" mode="instance"/>

      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PrimitiveBehaviorElemen
t" mode="instance">
            <xsl:param name="branchPath"/>
            <xsl:param name="postCondLoop"/>

            <beh:PrimitiveBehavior>
                  <!--External Template - Template found in timeline-
primitiveBehavior.xslt-->
                  <xsl:call-template name="PrimitiveBehaviorElementInst"/>

                  <!--External Template - Template found in variable.xslt-->
                  <xsl:apply-templates
select="mapping/*/*/net.onesaf.services.model.representation.behavior.Entry"
mode="reference">
                        <xsl:with-param name="behaviorName"
select="behaviorName"/>
                  </xsl:apply-templates>

                  <xsl:call-template name="nextComponent">
                        <xsl:with-param name="branchPath" select="$branchPath"/>
                        <xsl:with-param name="postCondLoop"
select="$postCondLoop"/>
                  </xsl:call-template>

            </beh:PrimitiveBehavior>

            <!--External Template - Template found in variable.xslt-->
            <xsl:apply-templates
select="mapping/*/*/net.onesaf.services.model.representation.behavior.Entry"
mode="instance">
                  <xsl:with-param name="behaviorName" select="behaviorName"/>
            </xsl:apply-templates>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.CompositeBehaviorElemen
t" mode="instance">
            <xsl:param name="branchPath"/>
            <xsl:param name="postCondLoop"/>

            <beh:CompositeBehavior>
                  <!--External Template - Template found in timeline-
compositeBehavior.xslt-->
                  <xsl:call-template name="CompositeBehaviorElementInst"/>
```

```
                <!--External Template - Template found in variable.xslt-->
                <xsl:apply-templates
select="mapping/*/*/net.onesaf.services.model.representation.behavior.Entry"
mode="reference">
                        <xsl:with-param name="behaviorName"
select="behaviorName"/>
                </xsl:apply-templates>

                <xsl:call-template name="nextComponent">
                        <xsl:with-param name="branchPath" select="$branchPath"/>
                        <xsl:with-param name="postCondLoop"
select="$postCondLoop"/>
                </xsl:call-template>

            </beh:CompositeBehavior>

            <!--External Template - Template found in variable.xslt-->
            <xsl:apply-templates
select="mapping/*/*/net.onesaf.services.model.representation.behavior.Entry"
mode="instance">
                <xsl:with-param name="behaviorName" select="behaviorName"/>
            </xsl:apply-templates>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.OrderSenderElement"
mode="instance">
            <xsl:param name="branchPath"/>
            <xsl:param name="postCondLoop"/>

            <beh:CompositeBehavior>
                <!--External Template - Template found in timeline-
orderSender.xslt-->
                <xsl:call-template name="OrderSenderElementInst"/>

                <!--External Template - Template found in variable.xslt-->
                <xsl:apply-templates
select="mapping/*/*/net.onesaf.services.model.representation.behavior.Entry"
mode="reference">
                        <xsl:with-param name="behaviorName"
select="behaviorName"/>
                </xsl:apply-templates>

                <xsl:call-template name="nextComponent">
                        <xsl:with-param name="branchPath" select="$branchPath"/>
                        <xsl:with-param name="postCondLoop"
select="$postCondLoop"/>
                </xsl:call-template>

            </beh:CompositeBehavior>

            <!--External Template - Template found in variable.xslt-->
            <xsl:apply-templates
select="mapping/*/*/net.onesaf.services.model.representation.behavior.Entry"
mode="instance">
                <xsl:with-param name="behaviorName" select="behaviorName"/>
```

```
            </xsl:apply-templates>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.ConditionalBranchElemen
t" mode="instance">
            <beh:ConditionalBranch>
                  <!--External Template - Template found in timeline-
conditionalBranch.xslt-->
                  <xsl:call-template name="CB-ConditionalExpressionRef"/>
                  <!--External Template - Template called depends on first child
of path. -->
                  <xsl:apply-templates select="falsePath/*[1]"
mode="falsePath"/>
                  <xsl:apply-templates select="truePath/*[1]" mode="truePath"/>
            </beh:ConditionalBranch>

            <!--External Template - Template found in timeline-
conditionalBranch.xslt-->
            <xsl:apply-templates
select="conditional/net.onesaf.services.model.representation.behavior.AtomicPred
icateElement" mode="instance"/>

            <!--External Template - Template found in variable.xslt-->
            <xsl:apply-templates
select="mapping/*/*/net.onesaf.services.model.representation.behavior.Entry"
mode="instance">
                  <xsl:with-param name="behaviorName" select="behaviorName"/>
            </xsl:apply-templates>

            <xsl:apply-templates select="falsePath/*[1]" mode="instance">
                  <xsl:with-param name="branchPath" select="following-
sibling::*[1]"/>
            </xsl:apply-templates>
            <xsl:apply-templates select="truePath/*[1]" mode="instance">
                  <xsl:with-param name="branchPath" select="following-
sibling::*[1]"/>
            </xsl:apply-templates>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PostConditionalLoopElem
ent" mode="instance">
            <xsl:apply-templates select="timelineElement/*[1]" mode="instance">
                  <xsl:with-param name="postCondLoop" select="@refID"/>
            </xsl:apply-templates>

            <beh:ConditionalBranch>
                  <!--External Template - Template found in timeline-
postConditionalLoop.xslt-->
                  <xsl:call-template name="PCL-ConditionalExpressionRef"/>

                  <!-- The next sibling component, if any, of the
PostConditionalLoopElement would be the FalseBranchPath "next component" for the
ConditionalBranch.-->
```

```
                  <!-- If the next sibling component were another
PostConditionalLoopElement, use its generated Container as the FalseBranchPath
next component.-->
                  <!-- If no next sibling component exists, there is no need to
instantiate the hasFalseBranchPath object type property.-->
                  <xsl:variable name="followingSib" select="following-
sibling::*[1]"/>
                  <xsl:if test="$followingSib">
                        <xsl:apply-templates select="$followingSib"
mode="falsePath"/>
                  </xsl:if>

                  <xsl:apply-templates select="timelineElement/*[1]"
mode="truePath"/>
            </beh:ConditionalBranch>

            <!--External Template - Possibilities Include:

      net.onesaf.services.model.representation.behavior.AtomicPredicateElement -
Template found in timeline-conditionalBranch.xslt

      net.onesaf.services.model.representation.behavior.BinarySentenceElement -
Template found in timeline-postConditionalLoop.xslt

      net.onesaf.services.model.representation.behavior.UnarySentenceElement -
Template found in timeline-postConditionalLoop.xslt -->
            <xsl:apply-templates select="loopConditional/*[1]" mode="instance">
                  <xsl:with-param name="postCondLoop" select="@refID"/>
            </xsl:apply-templates>
      </xsl:template>

      <xsl:template name="nextComponent">
            <xsl:param name="branchPath"/>
            <xsl:param name="postCondLoop"/>
            <xsl:choose>
            <!-- This condition handles potential next component from within a
ConditionalBranchElement. -->
                  <xsl:when test="$branchPath">
                        <xsl:apply-templates select="$branchPath"
mode="nextComponent"/>
                  </xsl:when>
                  <xsl:otherwise>
                        <xsl:variable name="followingSib" select="following-
sibling::*[1]"/>
                        <xsl:choose>
                              <xsl:when test="$followingSib">
                                    <xsl:apply-templates select="$followingSib"
mode="nextComponent"/>
                              </xsl:when>
                              <!-- This captures the last-next component for
post conditional loops which need to point to a conditional branch. -->
                              <xsl:otherwise>
                                    <xsl:if test="$postCondLoop">
                                          <beh:hasNextComponent>
                                                <xsl:attribute
name="rdf:resource">
```

```
                                                                    <xsl:value-of
select="concat('#ConditionalBranch-',$postCondLoop)"/>
                                                                </xsl:attribute>
                                                            </beh:hasNextComponent>
                                                        </xsl:if>
                                                    </xsl:otherwise>
                                                </xsl:choose>
                                            </xsl:otherwise>
                                        </xsl:choose>
                                    </xsl:template>

</xsl:stylesheet>
```

Attachment 15 - timeline-orderSender.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <!-- External XSLT that contain templates called/applied within this XSLT
-->
      <xsl:import href="variable.xslt"/>
      <xsl:import href="timeline-common.xslt"/>

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.OrderSenderElement"
mode="firstComponent">
            <beh:hasFirstComponent>
                  <xsl:call-template name="defineAttributeForOS"/>
            </beh:hasFirstComponent>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.OrderSenderElement"
mode="reference">
            <beh:hasCompositeBehavior>
                  <xsl:call-template name="defineAttributeForOS"/>
            </beh:hasCompositeBehavior>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.OrderSenderElement"
mode="nextComponent">
            <beh:hasNextComponent>
                  <xsl:call-template name="defineAttributeForOS"/>
            </beh:hasNextComponent>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.OrderSenderElement"
mode="truePath">
            <beh:hasTrueBranchPath>
                  <xsl:call-template name="defineAttributeForOS"/>
            </beh:hasTrueBranchPath>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.OrderSenderElement"
mode="falsePath">
            <beh:hasFalseBranchPath>
                  <xsl:call-template name="defineAttributeForOS"/>
            </beh:hasFalseBranchPath>
      </xsl:template>

      <xsl:template name="defineAttributeForOS">
            <xsl:attribute name="rdf:resource">
```

```
                  <xsl:value-of select="string('#')"/>
                  <!--External Template - Template found in timeline-
common.xslt-->
                  <xsl:call-template name="internalOrderSenderName">
                        <xsl:with-param name="behaviorType">
                              <xsl:choose>
                                    <xsl:when
test="contains(behaviorName,'composite')">composite</xsl:when>
                                    <xsl:when
test="contains(behaviorName,'primitive')">primitive</xsl:when>
                              </xsl:choose>
                        </xsl:with-param>
                        <xsl:with-param name="behaviorName"
select="behaviorName"/>
                  </xsl:call-template>
            </xsl:attribute>
      </xsl:template>

      <xsl:template name="OrderSenderElementInst">
            <xsl:variable name="behaviorName" select="behaviorName"/>

            <xsl:attribute name="rdf:ID">
                  <!--External Template - Template found in timeline-
common.xslt-->
                  <xsl:call-template name="internalOrderSenderName">
                        <!-- There is one order sender that's a primitive.  For
now we will deal with it manually. -->
                        <xsl:with-param name="behaviorType"
select="string('composite')"/>
                        <xsl:with-param name="behaviorName"
select="$behaviorName"/>
                  </xsl:call-template>
            </xsl:attribute>

            <beh:OrderSender>True</beh:OrderSender>

            <!-- If subordinateListName contains prefix then parse it out and
find the behavior refID that has an elementIdentifier that matches the prefix --
>
            <xsl:choose>

                  <xsl:when test="contains(subordinateListName,'):')">
                        <xsl:variable name="subordPrefix" select="substring-
before(subordinateListName,' (')"/>
                        <xsl:variable name="subordPrefixWithParen"
select="substring-before(subordinateListName,'::')"/>
                        <xsl:variable name="subordID"
select="//timelineElements/*[elementIdentifier=$subordPrefixWithParen]/@refID"/>
                        <beh:hasSubordinatesRetrievalPredicate>
                              <xsl:attribute name="rdf:resource">
                                    <xsl:value-of
select="concat('#',$subordPrefix,'-',$subordID)"/>
                              </xsl:attribute>
                        </beh:hasSubordinatesRetrievalPredicate>

                        <xsl:variable name="subordName" select="substring-
after(subordinateListName,'):')"/>
```

```
                        <xsl:variable name="subordFullName"
select="subordinateListName"/>
                        <xsl:variable name="behaviorName"
select="//timelineElements/*[elementIdentifier='planFireTeamDismount_PB
(1)']/behaviorName"/>
                        <xsl:variable name="fileLocation">
                              <xsl:value-of
select="concat('../oos/behaviors/',$behaviorName,'.xml')"/>
                        </xsl:variable>

                        <!--Need to go external document and figure out if
variable is local, input, or output then name accordingly -->
                        <xsl:variable name="extVariableType"
select="document($fileLocation)//net.onesaf.services.model.representation.behavi
or.TableEntry[@name=$subordName]/@variableType"/>

                         <xsl:variable name="variableType"
select="//net.onesaf.services.model.representation.behavior.TableEntry[@name=$su
bordFullName]/@variableType"/>
                        <beh:SubordinateListName>
                                    <xsl:choose>
                                          <xsl:when test="$variableType=0">
                                                <xsl:value-of
select="concat('LocalVar-',$subordName)"/>
                                          </xsl:when>
                                          <xsl:when test="$variableType=1">
                                                <xsl:value-of
select="concat('InputVar-',$subordName)"/>
                                          </xsl:when>
                                          <xsl:when test="$variableType=2">
                                                <xsl:value-of
select="concat('OutputVar-',$subordName)"/>
                                          </xsl:when>
                                    </xsl:choose>
                        </beh:SubordinateListName>
                  </xsl:when>

                  <xsl:otherwise>
                        <beh:SubordinateListName>
                              <xsl:value-of select="subordinateListName"/>
                        </beh:SubordinateListName>
                  </xsl:otherwise>

            </xsl:choose>

            <beh:usesBehaviorOf>
                  <!--External Template - Template found in timeline-
common.xslt-->
                  <xsl:call-template name="externalBehaviorName"/>
            </beh:usesBehaviorOf>
      </xsl:template>

</xsl:stylesheet>
```

Attachment 16 - timeline-postConditionalLoop.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <!-- External XSLT that contain templates called/applied within this XSLT
-->
      <xsl:import href="variable.xslt"/>
      <xsl:import href="timeline-common.xslt"/>
      <xsl:import href="timeline-primitiveBehavior.xslt"/>

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PostConditionalLoopElem
ent" mode="firstComponent">
            <xsl:variable name="firstTimelineChild"
select="name(timelineElement/*[1])"/>
            <xsl:choose>
                  <!-- If PCL has timelineElement/TemporalRelationElement child
-->
                  <xsl:when
test="contains($firstTimelineChild,'TemporalRelationElement')">
                        <beh:hasFirstComponent>
                              <xsl:call-template name="defineAttributeForPCL"/>
                        </beh:hasFirstComponent>
                  </xsl:when>
                  <!-- If PCL has timelineElement/PrimitiveBehaviorElement child
-->
                  <xsl:when
test="contains($firstTimelineChild,'PrimitiveBehaviorElement')">
                        <!-- Template defined in timeline-primitiveBehavior.xslt
-->
                        <xsl:apply-templates
select="timelineElement/net.onesaf.services.model.representation.behavior.Primit
iveBehaviorElement" mode="firstComponent"/>
                  </xsl:when>
            </xsl:choose>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PostConditionalLoopElem
ent" mode="reference">
            <xsl:apply-templates select="timelineElement/*[1]"
mode="reference"/>
            <beh:hasConditionalBranch>
                  <xsl:attribute name="rdf:resource">
                        <xsl:value-of select="concat('#ConditionalBranch-
',@refID)"/>
                  </xsl:attribute>
            </beh:hasConditionalBranch>
      </xsl:template>
```

```
      <xsl:template
match="net.onesaf.services.model.representation.behavior.PostConditionalLoopElem
ent" mode="nextComponent">
            <beh:hasNextComponent>
                  <xsl:call-template name="defineAttributeForPCL"/>
            </beh:hasNextComponent>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PostConditionalLoopElem
ent" mode="truePath">
            <beh:hasTrueBranchPath>
                  <xsl:call-template name="defineAttributeForPCL"/>
            </beh:hasTrueBranchPath>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PostConditionalLoopElem
ent" mode="falsePath">
            <beh:hasFalseBranchPath>
                  <xsl:call-template name="defineAttributeForPCL"/>
            </beh:hasFalseBranchPath>
      </xsl:template>

      <xsl:template name="defineAttributeForPCL">
            <xsl:attribute name="rdf:resource">
                  <xsl:value-of select="concat('#Container-
',timelineElement/*[1]/@refID)"/>
            </xsl:attribute>
      </xsl:template>

      <xsl:template name="PCL-ConditionalExpressionRef">
            <xsl:attribute name="rdf:ID">
                  <xsl:value-of select="concat('ConditionalBranch-',@refID)"/>
            </xsl:attribute>
            <xsl:variable name="loopConditionalChild"
select="name(loopConditional/*[1])"/>
            <beh:hasConditionalExpression>
                  <xsl:attribute name="rdf:resource">
                        <xsl:apply-templates select="*[1]/*[1]"
mode="reference"/>
                  </xsl:attribute>
            </beh:hasConditionalExpression>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.UnarySentenceElement"
mode="instance">
            <xsl:param name="postCondLoop"/>
            <xsl:variable name="unType" select="type"/>

            <beh:UnarySentence>
                  <xsl:attribute name="rdf:ID">
                        <xsl:value-of select="concat('UnarySentence-',@refID)"/>
                  </xsl:attribute>
                  <beh:hasUnaryOperatorType>
                        <xsl:attribute name="rdf:resource">
```

```
                              <xsl:choose>
                                    <xsl:when test="contains($unType,'not')">
                                          <xsl:value-of select="concat('#Not-
',@refID)"/>
                                    </xsl:when>
                              </xsl:choose>
                        </xsl:attribute>
                  </beh:hasUnaryOperatorType>
                  <beh:hasUnaryExpression>
                        <xsl:attribute name="rdf:resouce">
                              <xsl:apply-templates
select="logicalSentenceElement/*[1]" mode="reference"/>
                        </xsl:attribute>
                  </beh:hasUnaryExpression>
            </beh:UnarySentence>

            <xsl:choose>
                  <xsl:when test="$unType='not'">
                        <beh:UnaryNot>
                              <xsl:attribute name="rdf:ID">
                                    <xsl:value-of select="concat('Not-
',@refID)"/>
                              </xsl:attribute>
                              <beh:Note/>
                        </beh:UnaryNot>
                  </xsl:when>
            </xsl:choose>

            <!--External Template - Possibilities Include:

      net.onesaf.services.model.representation.behavior.AtomicPredicateElement -
Template found in timeline-conditionalBranch.xslt

      net.onesaf.services.model.representation.behavior.BinarySentenceElement -
Template found in timeline-postConditionalLoop.xslt

      net.onesaf.services.model.representation.behavior.UnarySentenceElement -
Template found in timeline-postConditionalLoop.xslt -->
            <xsl:apply-templates select="logicalSentenceElement/*[1]"
mode="instance"/>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.BinarySentenceElement"
mode="instance">
            <xsl:param name="postCondLoop"/>
            <xsl:variable name="binType" select="type"/>

            <beh:BinarySentence>
                  <xsl:attribute name="rdf:ID">
                        <xsl:value-of select="concat('BinarySentence-
',@refID)"/>
                  </xsl:attribute>
                  <beh:hasBinaryOperatorType>
                        <xsl:attribute name="rdf:resource">
                              <xsl:choose>
                                    <xsl:when test="$binType='AND'">
```

```
                                        <xsl:value-of
select="concat('#BinaryAnd-',@refID)"/>
                                    </xsl:when>
                                    <xsl:when test="$binType='OR'">
                                        <xsl:value-of
select="concat('#BinaryOr-',@refID)"/>
                                    </xsl:when>
                                </xsl:choose>
                            </xsl:attribute>
                        </beh:hasBinaryOperatorType>
                        <beh:hasRightHandSideExpression>
                            <xsl:attribute name="rdf:resource">
                                <xsl:apply-templates
select="rightHandSideLogicalSentence/*[1]" mode="reference"/>
                            </xsl:attribute>
                        </beh:hasRightHandSideExpression>
                        <beh:hasLeftHandSideExpression>
                            <xsl:attribute name="rdf:resource">
                                <xsl:apply-templates
select="leftHandSideLogicalSentence/*[1]" mode="reference"/>
                            </xsl:attribute>
                        </beh:hasLeftHandSideExpression>
                    </beh:BinarySentence>

                    <xsl:choose>
                        <xsl:when test="$binType='AND'">
                            <beh:BinaryAnd>
                                <xsl:attribute name="rdf:ID">
                                    <xsl:value-of select="concat('BinaryAnd-
',@refID)"/>
                                </xsl:attribute>
                                <beh:Note/>
                            </beh:BinaryAnd>
                        </xsl:when>
                        <xsl:when test="$binType='OR'">
                            <beh:BinaryOr>
                                <xsl:attribute name="rdf:ID">
                                    <xsl:value-of select="concat('BinaryOr-
',@refID)"/>
                                </xsl:attribute>
                                <beh:Note/>
                            </beh:BinaryOr>
                        </xsl:when>
                    </xsl:choose>

                    <!-- Right Hand Side Instance -->
                    <!--External Template - Possibilities Include:

        net.onesaf.services.model.representation.behavior.AtomicPredicateElement -
 Template found in timeline-conditionalBranch.xslt

        net.onesaf.services.model.representation.behavior.BinarySentenceElement -
 Template found in timeline-postConditionalLoop.xslt

        net.onesaf.services.model.representation.behavior.UnarySentenceElement -
 Template found in timeline-postConditionalLoop.xslt -->
```

```
            <xsl:apply-templates select="rightHandSideLogicalSentence/*[1]"
mode="instance"/>

            <!-- Left Hand Side Instance -->
            <!--External Template - Possibilities Include:

      net.onesaf.services.model.representation.behavior.AtomicPredicateElement -
Template found in timeline-conditionalBranch.xslt

      net.onesaf.services.model.representation.behavior.BinarySentenceElement -
Template found in timeline-postConditionalLoop.xslt

      net.onesaf.services.model.representation.behavior.UnarySentenceElement -
Template found in timeline-postConditionalLoop.xslt -->
            <xsl:apply-templates select="leftHandSideLogicalSentence/*[1]"
mode="instance"/>

      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.AtomicPredicateElement"
mode="reference">
            <xsl:value-of select="concat('#ConditionalExpression-',@refID)"/>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.UnarySentenceElement"
mode="reference">
            <xsl:value-of select="concat('#UnarySentence-',@refID)"/>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.BinarySentenceElement"
mode="reference">
            <xsl:value-of select="concat('#BinarySentence-',@refID)"/>
      </xsl:template>
</xsl:stylesheet>
```

Attachment 17 - timeline-primitiveBehavior.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <!-- External XSLT that contain templates called/applied within this XSLT
-->
      <xsl:import href="variable.xslt"/>
      <xsl:import href="timeline-common.xslt"/>

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PrimitiveBehaviorElemen
t" mode="firstComponent">
            <beh:hasFirstComponent>
                  <xsl:call-template name="defineAttributeForPBeh"/>
            </beh:hasFirstComponent>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PrimitiveBehaviorElemen
t" mode="reference">
            <beh:hasPrimitiveBehavior>
                  <xsl:call-template name="defineAttributeForPBeh"/>
            </beh:hasPrimitiveBehavior>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PrimitiveBehaviorElemen
t" mode="nextComponent">
            <beh:hasNextComponent>
                  <xsl:call-template name="defineAttributeForPBeh"/>
            </beh:hasNextComponent>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PrimitiveBehaviorElemen
t" mode="truePath">
            <beh:hasTrueBranchPath>
                  <xsl:call-template name="defineAttributeForPBeh"/>
            </beh:hasTrueBranchPath>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.PrimitiveBehaviorElemen
t" mode="falsePath">
            <beh:hasFalseBranchPath>
                  <xsl:call-template name="defineAttributeForPBeh"/>
            </beh:hasFalseBranchPath>
      </xsl:template>

      <xsl:template name="defineAttributeForPBeh">
            <xsl:attribute name="rdf:resource">
```

```
                  <xsl:value-of select="string('#')"/>
                  <!--External Template - Template found in timeline-
common.xslt-->
                  <xsl:call-template name="internalBehaviorName">
                        <xsl:with-param name="behaviorType"
select="string('primitive')"/>
                        <xsl:with-param name="behaviorName"
select="behaviorName"/>
                  </xsl:call-template>
            </xsl:attribute>
      </xsl:template>

      <xsl:template name="PrimitiveBehaviorElementInst">
            <xsl:variable name="behaviorName" select="behaviorName"/>

            <xsl:attribute name="rdf:ID">
                  <!--External Template - Template found in timeline-
common.xslt-->
                  <xsl:call-template name="internalBehaviorName">
                        <xsl:with-param name="behaviorType"
select="string('primitive')"/>
                        <xsl:with-param name="behaviorName"
select="$behaviorName"/>
                  </xsl:call-template>
            </xsl:attribute>

            <beh:usesBehaviorOf>
                  <!--External Template - Template found in timeline-
common.xslt-->
                  <xsl:call-template name="externalBehaviorName"/>
            </beh:usesBehaviorOf>
      </xsl:template>

      <xsl:template name="ConditionPrimitiveInstance">
            <xsl:param name="behaviorName"/>
            <xsl:param name="postCondLoop"/>
            <beh:PrimitiveBehavior>
                  <xsl:attribute name="rdf:ID">
                        <!--External Template - Template found in timeline-
common.xslt-->
                        <xsl:call-template name="internalBehaviorName">
                              <xsl:with-param name="behaviorType"
select="string('primitive')"/>
                              <xsl:with-param name="behaviorName"
select="$behaviorName"/>
                        </xsl:call-template>
                  </xsl:attribute>

                  <beh:usesBehaviorOf>
                        <xsl:attribute name="rdf:resource">
                              <xsl:choose>
                                    <xsl:when
test="contains($behaviorName,'.xml')">
                                          <!-- Use of $extURI which is defined
in main.xslt -->
                                          <xsl:value-of
select="concat($extURI,substring-after(substring-
```

```
before($behaviorName,'.xml'),'behavior/'),'.rdf#',translate(substring-
before($behaviorName,'.xml'),'/','.'))"/>
                                    </xsl:when>
                                    <xsl:otherwise>
                                          <!-- Use of $extURI which is defined
in main.xslt -->
                                          <xsl:value-of
select="concat($extURI,substring-
after($behaviorName,'behavior/'),'.rdf#',translate($behaviorName,'/','.'))"/>
                                    </xsl:otherwise>
                              </xsl:choose>
                        </xsl:attribute>
                  </beh:usesBehaviorOf>
                  <beh:Note/>
                  <!-- Add references to any input/output variables if any
mappings exist-->
                  <xsl:if test="$postCondLoop">
                        <!--External Template - Template found in variable.xslt-
->
                        <xsl:apply-templates
select="predicateMapping/*[1]/mappings/net.onesaf.services.model.representation.
behavior.Entry" mode="reference">
                              <xsl:with-param name="behaviorName"
select="predicateDescriptionName"/>
                        </xsl:apply-templates>
                  </xsl:if>
            </beh:PrimitiveBehavior>

            <xsl:if test="$postCondLoop">
                  <!--External Template - Template found in variable.xslt-->
                  <xsl:apply-templates
select="predicateMapping/*[1]/mappings/net.onesaf.services.model.representation.
behavior.Entry" mode="instance">
                        <xsl:with-param name="behaviorName"
select="predicateDescriptionName"/>
                  </xsl:apply-templates>
            </xsl:if>

      </xsl:template>

</xsl:stylesheet>
```

Attachment 18 - variable.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
      xmlns:var="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#"
      xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">

      <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="no"/>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.TableEntry"
mode="reference">
            <xsl:choose>
                  <xsl:when test="@variableType=1">
                        <var:hasBehaviorInput>
                              <xsl:attribute name="rdf:resource">
                                    <xsl:value-of select="concat('#InputVar-
',@name)"/>
                              </xsl:attribute>
                        </var:hasBehaviorInput>
                  </xsl:when>
                  <xsl:when test="@variableType=2">
                        <var:hasBehaviorOutput>
                              <xsl:attribute name="rdf:resource">
                                    <xsl:value-of select="concat('#OutputVar-
',@name)"/>
                              </xsl:attribute>
                        </var:hasBehaviorOutput>
                  </xsl:when>
            </xsl:choose>
      </xsl:template>

      <!-- The following template will only get called from within the template
instantiating a Top Level Container. -->
      <xsl:template
match="net.onesaf.services.model.representation.behavior.TableEntry"
mode="containerRefVars">
            <xsl:choose>
                  <xsl:when test="@variableType=0">
                        <var:hasLocalVariable>
                              <xsl:attribute name="rdf:resource">
                                    <xsl:choose>
                                          <xsl:when
test="contains(@name,')::')">
                                                <xsl:value-of
select="concat('#LocalVar-',translate(substring-after(@name,')::'),' ',''))"/>
                                          </xsl:when>
                                          <xsl:otherwise>
                                                <xsl:value-of
select="concat('#LocalVar-',@name)"/>
                                          </xsl:otherwise>
                                    </xsl:choose>
                              </xsl:attribute>
                        </var:hasLocalVariable>
                  </xsl:when>
                  <xsl:when test="@variableType=1">
```

```
                        <var:hasInput>
                              <xsl:attribute name="rdf:resource">
                                    <xsl:value-of select="concat('#InputVar-
',@name)"/>
                              </xsl:attribute>
                        </var:hasInput>
                  </xsl:when>
                  <xsl:when test="@variableType=2">
                        <var:hasOutput>
                              <xsl:attribute name="rdf:resource">
                                    <xsl:value-of select="concat('#OutputVar-
',@name)"/>
                              </xsl:attribute>
                        </var:hasOutput>
                  </xsl:when>
            </xsl:choose>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.TableEntry"
mode="instance">
            <xsl:choose>
                  <xsl:when test="@variableType=0">
                        <xsl:choose>
                              <xsl:when test="contains(@name,')::')">
                                    <!-- Need to make sure there isn't a
preceding sibling that hasn't already instantiated the current local variable --
>
                                    <xsl:variable name="localVarName"
select="translate(substring-after(@name,')::'),' ','')"/>
                                    <xsl:variable name="precedingVar"
select="preceding-sibling::node()[translate(substring-after(@name,')::'),'
','')=$localVarName]"/>
                                    <xsl:choose>
                                          <xsl:when test="not($precedingVar)">
                                                <var:LocalVariable>
                                                      <xsl:attribute
name="rdf:ID">
                                                            <xsl:value-of
select="concat('LocalVar-',$localVarName)"/>
                                                      </xsl:attribute>
                                                      <xsl:call-template
name="LocalInputOutputVariables"/>
                                                </var:LocalVariable>
                                          </xsl:when>
                                    </xsl:choose>
                              </xsl:when>
                              <xsl:otherwise>
                                    <var:LocalVariable>
                                          <xsl:attribute name="rdf:ID">
                                                <xsl:value-of
select="concat('LocalVar-',@name)"/>
                                          </xsl:attribute>
                                          <xsl:call-template
name="LocalInputOutputVariables"/>
                                    </var:LocalVariable>
                              </xsl:otherwise>
```

```
                        </xsl:choose>
                  </xsl:when>
                  <xsl:when test="@variableType=1">
                        <var:Input>
                              <xsl:attribute name="rdf:ID"><xsl:value-of
select="concat('InputVar-',@name)"/></xsl:attribute>
                              <xsl:call-template
name="LocalInputOutputVariables"/>
                        </var:Input>
                  </xsl:when>
                  <xsl:when test="@variableType=2">
                        <var:Output>
                              <xsl:attribute name="rdf:ID"><xsl:value-of
select="concat('OutputVar-',@name)"/></xsl:attribute>
                              <xsl:call-template
name="LocalInputOutputVariables"/>
                        </var:Output>
                  </xsl:when>
            </xsl:choose>
      </xsl:template>

      <xsl:template name="LocalInputOutputVariables">
            <xsl:choose>
                  <xsl:when test="contains(@name,')::')">
                        <xsl:variable name="behaviorName"
select="//timelineElements/*/behaviorName[contains(.,substring-before(@name,'
'))]"/>
                        <xsl:variable name="variableName"   select="substring-
after(@name,')::')"/>
                        <xsl:variable name="fileLocation">
                              <xsl:value-of
select="concat('../oos/behaviors/',$behaviorName,'.xml')"/>
                        </xsl:variable>
                        <xsl:variable name="variableType"
select="document($fileLocation)//net.onesaf.services.model.representation.behavi
or.TableEntry[@name=$variableName]/@variableType"/>
                        <var:Name>
                              <xsl:value-of select="$variableName"/>
                        </var:Name>
                        <var:DisplayName>
                                    <xsl:value-of select="substring-
after(@name,')::')"/>
                        </var:DisplayName>
                        <!--
                        <var:usesValueFrom>
                              <xsl:attribute name="rdf:resource">
                                    <xsl:choose>
                                          <xsl:when test="$variableType=0">
                                                <xsl:value-of
select="concat($extURI,substring-
after($behaviorName,'behavior/'),'.rdf#','LocalVar-',$variableName)"/>
                                          </xsl:when>
                                          <xsl:when test="$variableType=1">
                                                <xsl:value-of
select="concat($extURI,substring-
after($behaviorName,'behavior/'),'.rdf#','InputVar-',$variableName)"/>
                                          </xsl:when>
```

```
                                                <xsl:when test="$variableType=2">
                                                      <xsl:value-of
select="concat($extURI,substring-
after($behaviorName,'behavior/'),'.rdf#','OutputVar-',$variableName)"/>
                                                </xsl:when>
                                          </xsl:choose>
                                    </xsl:attribute>
                              </var:usesValueFrom>
                              -->
                        </xsl:when>
                        <xsl:otherwise>
                              <var:Name>
                                    <xsl:value-of select="@name"/>
                              </var:Name>
                              <var:DisplayName>
                                          <xsl:value-of select="@name"/>
                              </var:DisplayName>
                        </xsl:otherwise>
                  </xsl:choose>
                  <var:DataType>
                        <xsl:value-of select="@dataType"/>
                  </var:DataType>
                  <var:DefaultValue>
                        <xsl:value-of select="@defaultValue"/>
                  </var:DefaultValue>
                  <var:Required>
                        <xsl:value-of select="@mandatory"/>
                  </var:Required>
                  <var:Note>
                        <xsl:value-of select="concat('indexType=',@indexType)"/>
                  </var:Note>
            </xsl:template>

      <!-- Analysis of OOS behavior xml files shows that there are @rhs attributes for
      variables with names that seem to reference an external behavior file
                  (e.g., planFireTeamDismount_PB (1)::subordinates).  These attributes
      need to be handled seperately by testing with... contains(@rhs,')::')
      -->

            <xsl:template
      match="net.onesaf.services.model.representation.behavior.Entry"
      mode="reference">
                  <xsl:param name="behaviorName"/>

                  <xsl:variable name="fileLocation">
                        <xsl:value-of
      select="concat('../oos/behaviors/',$behaviorName,'.xml')"/>
                  </xsl:variable>
                  <xsl:variable name="lhs" select="@lhs"/>
                  <xsl:variable name="extVariableType"
      select="document($fileLocation)//net.onesaf.services.model.representation.behavi
      or.TableEntry[@name=$lhs]/@variableType"/>

                  <xsl:choose>
                        <!-- Mapping the input variables of @lhs. -->
                        <xsl:when test="$extVariableType=1">
                              <var:hasBehaviorInput>
```

```
                              <xsl:attribute name="rdf:resource">
                                    <!-- Use of $extURI which is defined in
main.xslt -->
                                    <xsl:value-of select="concat('#InputVar-
',@lhs,'-',@refID)"/>
                              </xsl:attribute>
                        </var:hasBehaviorInput>
                  </xsl:when>
                  <xsl:when test="$extVariableType=2">
                        <var:hasBehaviorOutput>
                              <xsl:attribute name="rdf:resource">
                                    <!-- Use of $extURI which is defined in
main.xslt -->
                                    <xsl:value-of select="concat('#OutputVar-
',@lhs,'-',@refID)"/>
                              </xsl:attribute>
                        </var:hasBehaviorOutput>
                  </xsl:when>
            </xsl:choose>
      </xsl:template>

      <xsl:template
match="net.onesaf.services.model.representation.behavior.Entry" mode="instance">
            <xsl:param name="behaviorName"/>

            <xsl:variable name="fileLocation">
                  <xsl:value-of
select="concat('../oos/behaviors/',$behaviorName,'.xml')"/>
            </xsl:variable>
            <xsl:variable name="lhs" select="@lhs"/>
            <xsl:variable name="extVariableType"
select="document($fileLocation)//net.onesaf.services.model.representation.behavi
or.TableEntry[@name=$lhs]/@variableType"/>
            <xsl:variable name="rhs" select="@rhs"/>
            <xsl:variable name="variableType"
select="//net.onesaf.services.model.representation.behavior.TableEntry[@name=$rh
s]/@variableType"/>
            <xsl:variable name="variableName" select="substring-
after(@rhs,')::')"/>

            <xsl:choose>
                  <!-- Mapping the input variables of @lhs. -->
                  <xsl:when test="$extVariableType=1">
                        <var:Input>
                              <xsl:attribute name="rdf:ID">
                                    <!-- Use of $extURI which is defined in
main.xslt -->
                                    <xsl:value-of select="concat('InputVar-
',@lhs,'-',@refID)"/>
                              </xsl:attribute>
                              <var:usesValueFrom>
                                    <xsl:attribute name="rdf:resource">
                                          <xsl:choose>
                                                <xsl:when
test="contains(@rhs,')::')">
                                                      <xsl:value-of
select="concat('#LocalVar-',translate($variableName,' ',''))"/>
```

```
                                                </xsl:when>
                                                <xsl:otherwise>
                                                      <xsl:choose>
                                                            <xsl:when
test="$variableType=0">
                                                                  <xsl:value-of
select="concat('#LocalVar-',$rhs)"/>
                                                            </xsl:when>
                                                            <xsl:when
test="$variableType=1">
                                                                  <xsl:value-of
select="concat('#InputVar-',$rhs)"/>
                                                            </xsl:when>
                                                            <xsl:when
test="$variableType=2">
                                                                  <xsl:value-of
select="concat('#OutputVar-',$rhs)"/>
                                                            </xsl:when>
                                                      </xsl:choose>
                                                </xsl:otherwise>
                                          </xsl:choose>
                                    </xsl:attribute>
                              </var:usesValueFrom>
                              <var:usedForValueOf>
                                    <xsl:attribute name="rdf:resource">
                                          <!-- Use of $extURI which is defined
in main.xslt -->
                                          <xsl:value-of
select="concat($extURI,substring-
after($behaviorName,'behavior/'),'.rdf#InputVar-',@lhs)"/>
                                    </xsl:attribute>
                              </var:usedForValueOf>
                        </var:Input>
                  </xsl:when>

                  <!-- Mapping the output variables of @lhs. -->
                  <xsl:when test="$extVariableType=2">
                        <var:Output>
                              <xsl:attribute name="rdf:ID">
                                    <!-- Use of $extURI which is defined in
main.xslt -->
                                    <xsl:value-of select="concat('OutputVar-
',@lhs,'-',@refID)"/>
                              </xsl:attribute>
                              <var:usesValueFrom>
                                    <xsl:attribute name="rdf:resource">
                                          <!-- Use of $extURI which is defined
in main.xslt -->
                                          <xsl:value-of
select="concat($extURI,substring-
after($behaviorName,'behavior/'),'.rdf#OutputVar-',@lhs)"/>
                                    </xsl:attribute>
                              </var:usesValueFrom>
                              <var:usedForValueOf>
                                    <xsl:attribute name="rdf:resource">
                                          <xsl:choose>
```

```
                                                <xsl:when
test="contains(@rhs,'):')">
                                                      <xsl:value-of
select="concat('#LocalVar-',translate($variableName,' ',''))"/>
                                                </xsl:when>
                                                <xsl:otherwise>
                                                      <xsl:choose>
                                                            <xsl:when
test="$variableType=0">
                                                                  <xsl:value-of
select="concat('#LocalVar-',translate($rhs,' ',''))"/>
                                                            </xsl:when>
                                                            <xsl:when
test="$variableType=1">
                                                                  <xsl:value-of
select="concat('#InputVar-',translate($rhs,' ',''))"/>
                                                            </xsl:when>
                                                            <xsl:when
test="$variableType=2">
                                                                  <xsl:value-of
select="concat('#OutputVar-',translate($rhs,' ',''))"/>
                                                            </xsl:when>
                                                      </xsl:choose>
                                                </xsl:otherwise>
                                          </xsl:choose>
                                    </xsl:attribute>
                              </var:usedForValueOf>
                        </var:Output>
                  </xsl:when>

            </xsl:choose>
      </xsl:template>

</xsl:stylesheet>
```

Attachment 19–JSAF vmove_task.fsm DRIVING params Behavior

```
DRIVING
    tick
                {
                    int32                   component =
cmpnt_locate(vehicle_id,
                                            reader_get_symbol("hull"));
                    if (!movement_enabled)
                  ^DISABLED; been shut down

                if (ent_get_cell(vehicle_id) != priv->cell)
                  vmove_update_private(vehicle_id, priv, state,
                                parameters, task_entry);

                vmove_tick_update(parameters, priv, state,
                              movement_enabled, unit_entry);

                /* Needed for AAAV */
                HULLS_SET_ANCHORED(vehicle_id, component, FALSE);

                if (vmove_heading_for_end(priv))
                  ^ARRIVING; we are heading for the end
                else if (vmove_stopped(priv))
                  ^STOPPED; we stopped
                else
                  ^DRIVING; keep going
            }

    params
                {
                if (parameters->termination & VMOVE_TERM_IMMEDIATE)
                  ^END; we were told to die immediately

                if (vmove_no_input_route(priv, parameters))
                  ^CHASING; make our own route

                vmove_update_private(vehicle_id, priv, state,
                              parameters, task_entry);
            }
```

NOTE: The params section is assumed to start with

```
DRIVING
```

when it is handled individually for the purpose of transforming the RDF/XML code back into JSAF like code.

Attachment 20–JSAF vmove_task.fsm START Behavior

```
START
    {
        priv->chase = FALSE;
        if (!state->cell)
          state->cell = wld_get_playbox_center_cell();
        vmove_update_private(vehicle_id, priv, state,
                             parameters, task_entry);
      ^DISABLED; always disabled when in start
    }
```

Attachment 21 - JSAF_vmove_DRIVING_params_CB.rdf

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
xmlns:var="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#"
xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
xmlns:cdm="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl#" xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">
      <beh:CompositeBehavior rdf:ID="JSAF_vmove_DRIVING_params_CB">
            <beh:DraftBehavior>True</beh:DraftBehavior>
            <beh:Resolution/>
            <beh:Fidelity/>
            <beh:hasTopLevelContainer rdf:resource="#Container-1"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-parameters-
_termination"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-
VMOVE_TERM_IMMEDIATE"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-priv"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-parameters"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-vehicle_id"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-state"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-task_entry"/>
            <beh:hasRulesPredicatesRepresentation
rdf:resource="#LocalReactionRule-2"/>
            <beh:hasRulesPredicatesRepresentation
rdf:resource="#LocalReactionRule-3"/>
            <art:hasGeneralMetadata rdf:resource="#GenMetadata-0"/>
            <art:hasCurrentVersion rdf:resource="#Ver-0"/>
            <cdm:hasConceptDomainMetadata
rdf:resource="#ConceptDomainMetadata-0"/>
      </beh:CompositeBehavior>
      <var:Input rdf:ID="InputVar-vehicle_id">
            <var:Name>vehicle_id</var:Name>
            <var:DisplayName>vehicle_id</var:DisplayName>
            <var:DataType>int32</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=0</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-task_entry">
            <var:Name>task_entry</var:Name>
            <var:DisplayName>task_entry</var:DisplayName>
            <var:DataType>PO_DB_ENTRY *</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=1</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-parameters">
            <var:Name>parameters</var:Name>
            <var:DisplayName>parameters</var:DisplayName>
            <var:DataType>VMOVE_PARAMETERS *</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=2</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-state">
            <var:Name>state</var:Name>
            <var:DisplayName>state</var:DisplayName>
            <var:DataType>VMOVE_STATE *</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=3</var:Note>
```

```
      </var:Input>
      <var:Input rdf:ID="InputVar-priv">
            <var:Name>priv</var:Name>
            <var:DisplayName>priv</var:DisplayName>
            <var:DataType>VMOVE_VARS *</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=4</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-parameters-_termination">
            <var:Name>parameters->termination</var:Name>
            <var:DisplayName>parameters->termination</var:DisplayName>
            <var:DataType>VMOVE_PARAMETERS->termination</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=5</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-VMOVE_TERM_IMMEDIATE">
            <var:Name>VMOVE_TERM_IMMEDIATE</var:Name>
            <var:DisplayName>VMOVE_TERM_IMMEDIATE</var:DisplayName>
            <var:DataType>Boolean</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=6</var:Note>
      </var:Input>
      <beh:Container rdf:ID="Container-1">
            <beh:ContainerTitle>Sequence: (1)</beh:ContainerTitle>

      <beh:ParallelContainerType>False</beh:ParallelContainerType>
            <beh:InBackground>False</beh:InBackground>
            <beh:hasFirstComponent rdf:resource="#Container-4"/>
            <beh:hasContainer rdf:resource="#Container-4"/>
            <beh:hasContainer rdf:resource="#Container-5"/>
            <beh:hasPrimitiveBehavior
rdf:resource="#JSAF_vmove_update_private_PB-6"/>
      </beh:Container>
      <beh:Container rdf:ID="Container-4">
            <beh:ContainerTitle>Sequence: (4)</beh:ContainerTitle>

      <beh:ParallelContainerType>False</beh:ParallelContainerType>
            <beh:InBackground>False</beh:InBackground>
            <beh:hasFirstComponent rdf:resource="#ConditionalBranch-
7"/>
            <beh:hasConditionalBranch rdf:resource="#ConditionalBranch-
7"/>
            <beh:hasPrimitiveBehavior
rdf:resource="#JSAF_AssignState_PB-8"/>
            <beh:hasNextComponent rdf:resource="#Container-5"/>
      </beh:Container>
      <beh:ConditionalBranch rdf:ID="ConditionalBranch-7">
            <beh:hasConditionalExpression
rdf:resource="#ConditionalExpression-11"/>
            <beh:hasTrueBranchPath rdf:resource="#JSAF_AssignState_PB-
8"/>
      </beh:ConditionalBranch>
      <beh:ConditionalExpression rdf:ID="ConditionalExpression-11">
            <beh:hasConditionalPredicate
rdf:resource="#JSAF_BitwiseAnd_PB-12"/>
      </beh:ConditionalExpression>
      <beh:PrimitiveBehavior rdf:ID="JSAF_BitwiseAnd_PB-12">
```

```
            <beh:usesBehaviorOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_BitwiseAnd_PB.rdf#JSAF_BitwiseAnd_PB"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-parameters-
_termination-13"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-
VMOVE_TERM_IMMEDIATE-14"/>
            <var:returnsValue rdf:resource="#OutputVar-result-15"/>
      </beh:PrimitiveBehavior>
      <var:Input rdf:ID="InputVar-parameters-_termination-13">
            <var:usesValueFrom rdf:resource="#InputVar-parameters-
_termination"/>
            <var:DisplayName>test-InputVar-parameters-
>termination</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_BitwiseAnd_PB.rdf#InputVar-leftHandSide"/>
            <var:Note>C_argument_number=0</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-VMOVE_TERM_IMMEDIATE-14">
            <var:usesValueFrom rdf:resource="#InputVar-
VMOVE_TERM_IMMEDIATE"/>
            <var:DisplayName>test-InputVar--
VMOVE_TERM_IMMEDIATE</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_BitwiseAnd_PB.rdf#InputVar-rightHandSide"/>
            <var:Note>C_argument_number=1</var:Note>
      </var:Input>
      <var:Output rdf:ID="OutputVar-result-15">
            <var:usesValueFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_BitwiseAnd_PB.rdf#OutputVar-result"/>
      </var:Output>
      <beh:PrimitiveBehavior rdf:ID="JSAF_AssignState_PB-8">
            <beh:usesBehaviorOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_AssignState_PB.rdf#JSAF_AssignState_PB"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-END-16"/>
            <beh:Note>we were told to die immediately</beh:Note>
      </beh:PrimitiveBehavior>
      <var:Input rdf:ID="InputVar-END-16">
            <var:Name>END</var:Name>
            <var:DisplayName>END</var:DisplayName>
            <var:DataType>enum</var:DataType>
            <var:Value>END</var:Value>
            <var:Required>True</var:Required>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_AssignState_PB.rdf#InputVar-newState"/>
            <var:Note>C_argument_number=0</var:Note>
      </var:Input>
      <beh:RulesPredicatesRepresentation rdf:ID="LocalReactionRule-2">
            <beh:GlobalReactionRule>False</beh:GlobalReactionRule>
            <beh:hasAntecedent rdf:resource="#ConditionalExpression-
11"/>
            <beh:effectiveAfter rdf:resource="#JSAF_AssignState_PB-8"/>
```

```
            <beh:hasConsequent
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/composite/JSAF
_vmove_END_CB.rdf#JSAF_vmove_END_CB"/>
      </beh:RulesPredicatesRepresentation>
      <beh:Container rdf:ID="Container-5">
            <beh:ContainerTitle>Sequence: (5)</beh:ContainerTitle>

      <beh:ParallelContainerType>False</beh:ParallelContainerType>
            <beh:InBackground>False</beh:InBackground>
            <beh:hasFirstComponent rdf:resource="#ConditionalBranch-
9"/>
            <beh:hasConditionalBranch rdf:resource="#ConditionalBranch-
9"/>
            <beh:hasPrimitiveBehavior
rdf:resource="#JSAF_AssignState_PB-10"/>
            <beh:hasNextComponent
rdf:resource="#JSAF_vmove_update_private_PB-6"/>
      </beh:Container>
      <beh:ConditionalBranch rdf:ID="ConditionalBranch-9">
            <beh:hasConditionalExpression
rdf:resource="#ConditionalExpression-17"/>
            <beh:hasTrueBranchPath rdf:resource="#JSAF_AssignState_PB-
10"/>
      </beh:ConditionalBranch>
      <beh:ConditionalExpression rdf:ID="ConditionalExpression-17">
            <beh:hasConditionalPredicate
rdf:resource="#JSAF_vmove_no_input_route_PB-18"/>
      </beh:ConditionalExpression>
      <beh:PrimitiveBehavior rdf:ID="JSAF_vmove_no_input_route_PB-18">
            <beh:usesBehaviorOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_no_input_route_PB.rdf#JSAF_vmove_no_input_route_PB"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-priv-19"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-parameters-
20"/>
            <var:returnsValue rdf:resource="#OutputVar-result-21"/>
      </beh:PrimitiveBehavior>
      <var:Input rdf:ID="InputVar-priv-19">
            <var:usesValueFrom rdf:resource="#InputVar-priv"/>
            <var:DisplayName>test-19</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_no_input_route_PB.rdf#InputVar-priv"/>
            <var:Note>C_argument_number=0</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-parameters-20">
            <var:usesValueFrom rdf:resource="#InputVar-parameters"/>
            <var:DisplayName>test-20</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_no_input_route_PB.rdf#InputVar-parameters"/>
            <var:Note>C_argument_number=1</var:Note>
      </var:Input>
      <var:Output rdf:ID="OutputVar-result-21">
            <var:usesValueFrom
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_no_input_route_PB.rdf#OutputVar-result"/>
```

```
            <var:DisplayName>test-21</var:DisplayName>
      </var:Output>
      <beh:PrimitiveBehavior rdf:ID="JSAF_AssignState_PB-10">
            <beh:usesBehaviorOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_AssignState_PB.rdf#JSAF_AssignState_PB"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-CHASING-22"/>
            <beh:Note>make our own route</beh:Note>
      </beh:PrimitiveBehavior>
      <var:Input rdf:ID="InputVar-CHASING-22">
            <var:Name>CHASING</var:Name>
            <var:DisplayName>CHASING</var:DisplayName>
            <var:DataType>enum</var:DataType>
            <var:Value>CHASING</var:Value>
            <var:Required>True</var:Required>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_AssignState_PB.rdf#InputVar-newState"/>
            <var:Note>C_argument_number=0</var:Note>
      </var:Input>
      <beh:RulesPredicatesRepresentation rdf:ID="LocalReactionRule-3">
            <beh:GlobalReactionRule>False</beh:GlobalReactionRule>
            <beh:hasAntecedent rdf:resource="#ConditionalExpression-
17"/>
            <beh:effectiveAfter rdf:resource="#JSAF_AssignState_PB-
10"/>
            <beh:hasConsequent
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/composite/JSAF
_vmove_CHASING_tick_CB.rdf#JSAF_vmove_CHASING_tick_CB"/>
      </beh:RulesPredicatesRepresentation>
      <beh:PrimitiveBehavior rdf:ID="JSAF_vmove_update_private_PB-6">
            <beh:usesBehaviorOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#JSAF_vmove_update_private_PB"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-vehicle_id-
23"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-priv-24"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-state-25"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-parameters-
26"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-task_entry-
27"/>
      </beh:PrimitiveBehavior>
      <var:Input rdf:ID="InputVar-vehicle_id-23">
            <var:usesValueFrom rdf:resource="#InputVar-vehicle_id"/>
            <var:DisplayName>test-23</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#InputVar-vehicle_id"/>
            <var:Note>C_argument_number=0</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-priv-24">
            <var:usesValueFrom rdf:resource="#InputVar-priv"/>
            <var:DisplayName>test-24</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#InputVar-priv"/>
```

```
            <var:Note>C_argument_number=1</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-state-25">
            <var:usesValueFrom rdf:resource="#InputVar-state"/>
            <var:DisplayName>test-25</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#InputVar-state"/>
            <var:Note>C_argument_number=2</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-parameters-26">
            <var:usesValueFrom rdf:resource="#InputVar-parameters"/>
            <var:DisplayName>test-26</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#InputVar-parameters"/>
            <var:Note>C_argument_number=3</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-task_entry-27">
            <var:usesValueFrom rdf:resource="#InputVar-task_entry"/>
            <var:DisplayName>test-27</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#InputVar-task_entry"/>
            <var:Note>C_argument_number=4</var:Note>
      </var:Input>
      <art:GeneralMetadata rdf:ID="GenMetadata-0">
            <art:Title/>
            <art:Subject/>
            <art:SecurityClassification>Official Use
Only</art:SecurityClassification>
            <art:Releasability>U.S. Government
Contractors</art:Releasability>
            <art:Description>null</art:Description>
      </art:GeneralMetadata>
      <art:Version rdf:ID="Ver-0">
            <art:VersionDate>2004-11/18</art:VersionDate>
            <art:hasVVARecord rdf:resource="#VVARec-0"/>
      </art:Version>
      <art:VVARecord rdf:ID="VVARec-0">
            <art:VerificationComplete>False</art:VerificationComplete>
            <art:ValidationComplete>False</art:ValidationComplete>

      <art:AccreditationComplete>False</art:AccreditationComplete>
            <art:hasVerificationAuthority rdf:resource="#Auth-0"/>
            <art:hasValidationAuthority rdf:resource="#Auth-0"/>
            <art:hasAccreditationAuthority rdf:resource="#Auth-0"/>
      </art:VVARecord>
      <art:Authority rdf:ID="Auth-0">
            <art:PersonName/>
            <art:PositionTitle/>
            <art:OrganizationName/>
      </art:Authority>
      <cdm:ConceptDomainMetadata rdf:ID="ConceptDomainMetadata-0">
            <cdm:hasAcceptableSide rdf:resource="#Side-1"/>
            <cdm:hasAcceptableEntityType rdf:resource="#Entity-1"/>
      </cdm:ConceptDomainMetadata>
```

```
        <cdm:Side rdf:ID="Side-1">
              <cdm:SideName>ANY</cdm:SideName>
        </cdm:Side>
        <cdm:Entity rdf:ID="Entity-1">
              <cdm:EntityType>ANY</cdm:EntityType>
        </cdm:Entity>
</rdf:RDF>
```

Attachment 22 - JSAF_vmove_START_CB.rdf

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
xmlns:var="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#"
xmlns:art="http://orlando.drc.com/hbr/Ontologies/Artifact-ont.owl#"
xmlns:cdm="http://orlando.drc.com/hbr/Ontologies/ConceptDomainMetadata-
ont.owl#" xmlns="http://orlando.drc.com/hbr/Artifacts/Behavior.owl#">
      <beh:CompositeBehavior rdf:ID="JSAF_vmove_START_CB">
            <beh:DraftBehavior>True</beh:DraftBehavior>
            <beh:Resolution/>
            <beh:Fidelity/>
            <beh:hasTopLevelContainer rdf:resource="#Container-1"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-vehicle_id"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-priv"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-state"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-parameters"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-task_entry"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-state-
_cell"/>
            <var:hasBehaviorOutput rdf:resource="#OutputVar-priv-
_chase"/>
            <beh:hasRulesPredicatesRepresentation
rdf:resource="#LocalReactionRule-2"/>
            <art:hasGeneralMetadata rdf:resource="#GenMetadata-0"/>
            <art:hasCurrentVersion rdf:resource="#Ver-0"/>
            <cdm:hasConceptDomainMetadata
rdf:resource="#ConceptDomainMetadata-0"/>
      </beh:CompositeBehavior>
      <var:Input rdf:ID="InputVar-vehicle_id">
            <var:Name>vehicle_id</var:Name>
            <var:DisplayName>vehicle_id</var:DisplayName>
            <var:DataType>int32</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=0</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-priv">
            <var:Name>priv</var:Name>
            <var:DisplayName>priv</var:DisplayName>
            <var:DataType>VMOVE_VARS *</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=1</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-state">
            <var:Name>state</var:Name>
            <var:DisplayName>state</var:DisplayName>
            <var:DataType>VMOVE_STATE *</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=2</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-parameters">
            <var:Name>parameters</var:Name>
            <var:DisplayName>parameters</var:DisplayName>
            <var:DataType>VMOVE_PARAMETERS *</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=3</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-task_entry">
```

```
            <var:Name>task_entry</var:Name>
            <var:DisplayName>task_entry</var:DisplayName>
            <var:DataType>PO_DB_ENTRY *</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=4</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-state-_cell">
            <var:Name>state->cell</var:Name>
            <var:DisplayName>state->cell</var:DisplayName>
            <var:DataType>VMOVE_STATE *</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=5</var:Note>
      </var:Input>
      <var:Output rdf:ID="OutputVar-priv-_chase">
            <var:Name>priv->chase</var:Name>
            <var:DisplayName>priv->chase</var:DisplayName>
            <var:DataType>VMOVE_STATE *</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=6</var:Note>
      </var:Output>
      <beh:Container rdf:ID="Container-1">
            <beh:ContainerTitle>Sequence: (1)</beh:ContainerTitle>

      <beh:ParallelContainerType>False</beh:ParallelContainerType>
            <beh:InBackground>False</beh:InBackground>
            <beh:hasFirstComponent rdf:resource="#JSAF_AssignValue_PB-
3"/>
            <beh:hasPrimitiveBehavior
rdf:resource="#JSAF_AssignValue_PB-3"/>
            <beh:hasContainer rdf:resource="#Container-4"/>
            <beh:hasPrimitiveBehavior
rdf:resource="#JSAF_vmove_update_private_PB-5"/>
            <beh:hasPrimitiveBehavior
rdf:resource="#JSAF_AssignState_PB-6"/>
      </beh:Container>
      <beh:PrimitiveBehavior rdf:ID="JSAF_AssignValue_PB-3">
            <beh:usesBehaviorOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_AssignValue_PB.rdf#JSAF_AssignValue_PB"/>
            <var:hasBehaviorOutput rdf:resource="#OutputVar-priv-
_chase-13"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-False-17"/>
            <beh:hasNextComponent rdf:resource="#Container-4"/>
      </beh:PrimitiveBehavior>
      <var:Output rdf:ID="OutputVar-priv-_chase-13">
            <var:usedForValueOf rdf:resource="#OutputVar-priv-_chase"/>
            <var:Note>C_argument_number=0</var:Note>
      </var:Output>
      <var:Input rdf:ID="InputVar-False-17">
            <var:Name>False</var:Name>
            <var:DisplayName>False</var:DisplayName>
            <var:DataType>boolean</var:DataType>
            <var:Required>True</var:Required>
            <var:Note>C_argument_number=1</var:Note>
      </var:Input>
      <beh:Container rdf:ID="Container-4">
            <beh:ContainerTitle>Sequence: (4)</beh:ContainerTitle>
```

```
      <beh:ParallelContainerType>False</beh:ParallelContainerType>
            <beh:InBackground>False</beh:InBackground>
            <beh:hasFirstComponent rdf:resource="#ConditionalBranch-
7"/>
            <beh:hasConditionalBranch rdf:resource="#ConditionalBranch-
7"/>
            <beh:hasPrimitiveBehavior
rdf:resource="#JSAF_AssignFunctionValue_PB-8"/>
            <beh:hasNextComponent
rdf:resource="#JSAF_vmove_update_private_PB-5"/>
      </beh:Container>
      <beh:ConditionalBranch rdf:ID="ConditionalBranch-7">
            <beh:hasConditionalExpression rdf:resource="#UnarySentence-
11"/>
            <beh:hasTrueBranchPath
rdf:resource="#JSAF_AssignFunctionValue_PB-8"/>
      </beh:ConditionalBranch>
      <beh:UnarySentence rdf:ID="UnarySentence-11">
            <beh:hasUnaryOperatorType rdf:resource="#Not-12"/>
            <beh:hasUnaryExpression
rdf:resource="#ConditionalExpression-11"/>
      </beh:UnarySentence>
      <beh:Not rdf:ID="Not-12">
      </beh:Not>
      <beh:ConditionalExpression rdf:ID="ConditionalExpression-11">
            <var:hasInput rdf:resource="#InputVar-state-_cell-13"/>
      </beh:ConditionalExpression>
      <var:Input rdf:ID="InputVar-state-_cell-13">
            <var:usesValueFrom rdf:resource="#InputVar-state-_cell"/>
      </var:Input>

      <beh:PrimitiveBehavior rdf:ID="JSAF_AssignFunctionValue_PB-8">
            <beh:usesBehaviorOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_AssignFunctionValue_PB.rdf#JSAF_AssignFunctionValue_PB"/>
            <var:hasBehaviorOutput rdf:resource="#OutputVar-state-
_cell-23"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-
JSAF_wld_get_playbox_center_cell_PB-24"/>
      </beh:PrimitiveBehavior>

      <var:Output rdf:ID="OutputVar-state-_cell-23">
            <var:usedForValueOf rdf:resource="#InputVar-state-_cell-
13"/>
            <var:Note>C_argument_number=0</var:Note>
      </var:Output>

      <var:Input rdf:ID="InputVar-JSAF_wld_get_playbox_center_cell_PB-
24">
            <var:Name>JSAF_wld_get_playbox_center_cell_PB</var:Name>
            <var:DisplayName>wld_get_playbox_center_cell(
)</var:DisplayName>
            <var:DataType>function</var:DataType>
            <var:Required>True</var:Required>

      <var:Source>http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/
```

```
JJSAF_wld_get_playbox_center_cell_PB.rdf#JSAF_wld_get_playbox_center_ce
ll_PB</var:Source>
            <var:Note>C_argument_number=1, </var:Note>
      </var:Input>

      <beh:PrimitiveBehavior rdf:ID="JSAF_vmove_update_private_PB-5">
            <beh:usesBehaviorOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#JSAF_vmove_update_private_PB"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-vehicle_id-
23"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-priv-24"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-state-25"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-parameters-
26"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-task_entry-
27"/>
            <beh:hasNextComponent rdf:resource="#JSAF_AssignState_PB-
6"/>
      </beh:PrimitiveBehavior>
      <var:Input rdf:ID="InputVar-vehicle_id-23">
            <var:usesValueFrom rdf:resource="#InputVar-vehicle_id"/>
            <var:DisplayName>test-23</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#InputVar-vehicle_id"/>
            <var:Note>C_argument_number=0</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-priv-24">
            <var:usesValueFrom rdf:resource="#InputVar-priv"/>
            <var:DisplayName>test-24</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#InputVar-priv"/>
            <var:Note>C_argument_number=1</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-state-25">
            <var:usesValueFrom rdf:resource="#InputVar-state"/>
            <var:DisplayName>test-25</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#InputVar-state"/>
            <var:Note>C_argument_number=2</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-parameters-26">
            <var:usesValueFrom rdf:resource="#InputVar-parameters"/>
            <var:DisplayName>test-26</var:DisplayName>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#InputVar-parameters"/>
            <var:Note>C_argument_number=3</var:Note>
      </var:Input>
      <var:Input rdf:ID="InputVar-task_entry-27">
            <var:usesValueFrom rdf:resource="#InputVar-task_entry"/>
            <var:DisplayName>test-27</var:DisplayName>
```

```
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_vmove_update_private_PB.rdf#InputVar-task_entry"/>
            <var:Note>C_argument_number=4</var:Note>
      </var:Input>
      <beh:PrimitiveBehavior rdf:ID="JSAF_AssignState_PB-6">
            <beh:usesBehaviorOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_AssignState_PB.rdf#JSAF_AssignState_PB"/>
            <var:hasBehaviorInput rdf:resource="#InputVar-DISABLED-
16"/>
            <beh:Note>always disabled when in start</beh:Note>
      </beh:PrimitiveBehavior>
      <var:Input rdf:ID="InputVar-DISABLED-16">
            <var:Name>DISABLED</var:Name>
            <var:DisplayName>DISABLED</var:DisplayName>
            <var:DataType>enum</var:DataType>
            <var:Value>DISABLED</var:Value>
            <var:Required>True</var:Required>
            <var:usedForValueOf
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/primitive/JSAF
_AssignState_PB.rdf#InputVar-newState"/>
            <var:Note>C_argument_number=0</var:Note>
      </var:Input>
      <beh:RulesPredicatesRepresentation rdf:ID="LocalReactionRule-2">
            <beh:GlobalReactionRule>False</beh:GlobalReactionRule>
            <beh:hasAntecedent rdf:resource="#UnarySentence-11"/>
            <beh:effectiveAfter rdf:resource="#JSAF_AssignState_PB-6"/>
            <beh:hasConsequent
rdf:resource="http://orlando.drc.com/hbr/Ontologies/JSAF/composite/JSAF
_vmove_DISABLED_CB.rdf#JSAF_vmove_DISABLED_CB"/>
      </beh:RulesPredicatesRepresentation>
      <art:GeneralMetadata rdf:ID="GenMetadata-0">
            <art:Title/>
            <art:Subject/>
            <art:SecurityClassification>Official Use
Only</art:SecurityClassification>
            <art:Releasability>U.S. Government
Contractors</art:Releasability>
            <art:Description>null</art:Description>
      </art:GeneralMetadata>
      <art:Version rdf:ID="Ver-0">
            <art:VersionDate>2004-11/18</art:VersionDate>
            <art:hasVVARecord rdf:resource="#VVARec-0"/>
      </art:Version>
      <art:VVARecord rdf:ID="VVARec-0">
            <art:VerificationComplete>False</art:VerificationComplete>
            <art:ValidationComplete>False</art:ValidationComplete>

      <art:AccreditationComplete>False</art:AccreditationComplete>
            <art:hasVerificationAuthority rdf:resource="#Auth-0"/>
            <art:hasValidationAuthority rdf:resource="#Auth-0"/>
            <art:hasAccreditationAuthority rdf:resource="#Auth-0"/>
      </art:VVARecord>
      <art:Authority rdf:ID="Auth-0">
            <art:PersonName/>
            <art:PositionTitle/>
```

```
            <art:OrganizationName/>
        </art:Authority>
        <cdm:ConceptDomainMetadata rdf:ID="ConceptDomainMetadata-0">
            <cdm:hasAcceptableSide rdf:resource="#Side-1"/>
            <cdm:hasAcceptableEntityType rdf:resource="#Entity-1"/>
        </cdm:ConceptDomainMetadata>
        <cdm:Side rdf:ID="Side-1">
            <cdm:SideName>ANY</cdm:SideName>
        </cdm:Side>
        <cdm:Entity rdf:ID="Entity-1">
            <cdm:EntityType>ANY</cdm:EntityType>
        </cdm:Entity>
</rdf:RDF>
```

Attachment 23 - JSAF_main.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:beh="http://orlando.drc.com/hbr/Ontologies/Behavior-ont.owl#"
xmlns:var="http://orlando.drc.com/hbr/Ontologies/Variable-ont.owl#">
      <xsl:output method="text" version="1.0" encoding="UTF-8"
indent="yes" omit-xml-declaration="yes"/>
    <xsl:variable name="globalName" select="substring-after(substring-
before(//beh:CompositeBehavior[beh:hasTopLevelContainer]/@rdf:ID,
'_CB'),'JSAF_vmove_')"/>
    <xsl:variable name="globalName2"
select="//beh:CompositeBehavior[beh:hasTopLevelContainer]/@rdf:ID"/>

<!-- This XSLT assumes a naming convention for JSAF behaviors presented
in an RDF/XML file.  The first portion will always be "JSAF_".  If the
function is global in scope, i.e., not specifically limited to
behaviors or functions in a specific behavior related library, it will
have the next portion of the name with capitals and lower case and will
end with PB for Primitive Behavior, e.g., "JSAF_AssignState_PB" or
"JSAF_AssignValue_PB".  If the function is limited in scope to a
specific library or associated specifically to the library, it will
have the second portion with that library name in lower case, e.g.,
"JSAF_vmove_" for libvmove, followed by the descriptive name with
underline separations and will end with "PB" for Primitive Behavior,
e.g., "JSAF_vmove_update_private_PB" or "JSAF_vmove_no_input_route_PB".
For the finite-state-machine state behavior names, which are associated
with specific libraries, the state name portion will be in all capital
letters followed, when applicable, by lower case "_tick_" or
"_params_", and end with "CB" for Composite Behavior, e.g.,
JSAF_vmove_START_CB or JSAF_vmove_DRIVE_params_CB.

Each specific composite and primitive behavior  are placed in their own
individual file using the name of the behavior followed by a ".rdf"
suffix, e.g., the "JSAF_vmove_DRIVE_params_CB" behavior is located in
file "JSAF_vmove_DRIVE_params_CB.rdf".

Since the behavior ontologies were not specifically designed for
writing code, normally used functions within code writing, such as "="
for assigning a value, are handled within the behavior ontologies as
primitive behaviors, e.g., JSAF_AssignValue_PB, with assumed inputs and
outputs.  Those primitive behavior substitutes are then decoded by the
XSLT to reproduce the function in its normally used form.

As an overview of the behavior ontology structure, containers are
temporal control structures that control the execution of the container
components within it that are "siblings" of each other at the same
level within the container.  Container components can be (1) other
containers, (2) conditional branches, (3) primitive behaviors, or (4)
composite behaviors.  Each container indicates which container
component will execute first within itself.  Each container conponent
also indicates which other container component, within the same
container, will next execute.  The conditional branch has one or two
paths, one for when the conditional expression for the conditional
branch evaluates to true and, possibly, though not always, a second
path for when the evaluation is false.  (The conditional branch
component allows branching and looping.)  The last component within a
```

```
container does not indicate a following component.  Container
components specify sequential or parallel execution of its components.
-->

<!-- Selects the root node in rdf:RDF -->
      <xsl:template match="/">
            <xsl:apply-templates select="rdf:RDF"/>
      </xsl:template>

<!-- Applies template to get to the top level Composite Behavior, which
will always be present. -->

      <xsl:template match="rdf:RDF">
            <xsl:apply-templates
select="beh:CompositeBehavior[beh:hasTopLevelContainer]"
mode="TopLevel"/>
      </xsl:template>

<!-- Determines the type of behavior to set as the header data for the
type of Composite Behavior, e.g., START, END, or state name and tick or
params. then applies template to the top level Container -->

      <xsl:template match="beh:CompositeBehavior" mode="TopLevel">

            <xsl:choose>
                  <xsl:when test="$globalName='START'">
                        <xsl:call-template name="outputSTART"/>
                  </xsl:when>
                  <xsl:when test="$globalName='END'">
                        <xsl:call-template name="outputEND"/>
                  </xsl:when>
                  <xsl:otherwise>
                        <xsl:choose>
                              <xsl:when
test="contains($globalName,'tick')">
                                    <xsl:call-template
name="outputTick"/>
                              </xsl:when>
                              <xsl:when
test="contains($globalName,'params')">
                                    <xsl:call-template
name="outputParams"/>
                              </xsl:when>
                              <xsl:otherwise>/* Top-level behavior name
is <xsl:value-of select="$globalName2"/> */
                              </xsl:otherwise>
                        </xsl:choose>
                  </xsl:otherwise>
            </xsl:choose>
            <xsl:variable name="containerID" select="substring-
after(beh:hasTopLevelContainer/@rdf:resource,'#')"/>
            <xsl:apply-templates
select="//beh:Container[@rdf:ID=$containerID]" mode="TopLevel"/>
      </xsl:template>

<!-- Template for evaluating the first component in the top level
container -->
```

```
      <xsl:template match="beh:Container" mode="TopLevel">
            {
                  <xsl:variable name="componentID" select="substring-
after(beh:hasFirstComponent/@rdf:resource,'#')"/>
                  <xsl:apply-templates
select="//*[@rdf:ID=$componentID]"/>
            }
      </xsl:template>

<!-- Applies template for the first component in a lower level
container and its next sibling component, if any. -->

      <xsl:template match="beh:Container">

                  <xsl:variable name="componentID" select="substring-
after(beh:hasFirstComponent/@rdf:resource,'#')"/>
                  <xsl:apply-templates
select="//*[@rdf:ID=$componentID]"/>
                  <xsl:variable name="componentIDNext"
select="substring-after(beh:hasNextComponent/@rdf:resource,'#')"/>
                  <xsl:if test="$componentIDNext">
                        <xsl:apply-templates
select="//*[@rdf:ID=$componentIDNext]"/>
                  </xsl:if>

      </xsl:template>

<!-- Applies template for type of Primitive Behavior and its next
sibling component, if any. -->

      <xsl:template match="beh:PrimitiveBehavior">
            <xsl:variable name="PrimitiveBehaviorIDName"
select="@rdf:ID"/>
            <xsl:choose>
                  <xsl:when
test="contains($PrimitiveBehaviorIDName,'JSAF_vmove_')">
                        <xsl:value-of select="substring-
before(substring-after(@rdf:ID,'JSAF_'),'_PB-')"/>(<xsl:apply-templates
select="var:hasBehaviorInput |
var:hasBehaviorOutput"/><xsl:text>);</xsl:text>
                  </xsl:when>
                  <xsl:otherwise>
                        <xsl:choose>
                              <xsl:when
test="contains($PrimitiveBehaviorIDName,'AssignState')">
                                    <xsl:call-template
name="outputState"/>
                              </xsl:when>
                              <xsl:when
test="contains($PrimitiveBehaviorIDName,'BitwiseAnd')">
                                    <xsl:call-template
name="outputBitwiseAnd"/>
                              </xsl:when>
                              <xsl:when
test="contains($PrimitiveBehaviorIDName,'AssignValue')">
```

```
                                        <xsl:call-template
name="outputAssignValue"/><xsl:text>;
                                    </xsl:text>
                                    </xsl:when>
                                    <xsl:when
test="contains($PrimitiveBehaviorIDName,'AssignFunctionValue')">
                                        <xsl:call-template
name="outputAssignFunctionValue"/><xsl:text>;
                                    </xsl:text>
                                    </xsl:when>
                                    <xsl:otherwise>
                                        <xsl:value-of
select="@rdf:ID"/>(<xsl:apply-templates select="var:hasBehaviorInput |
var:hasBehaviorOutput"/><xsl:text>);
                                    </xsl:text>
                                    </xsl:otherwise>
                                </xsl:choose>
                            </xsl:otherwise>
                            </xsl:choose>
                            <xsl:variable name="componentIDNext"
select="substring-after(beh:hasNextComponent/@rdf:resource,'#')"/>
                            <xsl:if test="$componentIDNext"><xsl:text>
                            </xsl:text>
                                <xsl:apply-templates
select="//*[@rdf:ID=$componentIDNext]"/>
                            </xsl:if>
        </xsl:template>

<!-- Applies template for type of Primitive Behavior when used in a
ConditionalExpression. -->

        <xsl:template match="beh:PrimitiveBehavior" mode="Conditional">
            <xsl:variable name="PrimitiveBehaviorIDName"
select="@rdf:ID"/>
            <xsl:choose>
                <xsl:when
test="contains($PrimitiveBehaviorIDName,'JSAF_vmove_')">
                    <xsl:value-of select="substring-
before(substring-after(@rdf:ID,'JSAF_'),'_PB-')"/>(<xsl:apply-templates
select="var:hasBehaviorInput |
var:hasBehaviorOutput"/><xsl:text>)</xsl:text>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:choose>
                        <xsl:when
test="contains($PrimitiveBehaviorIDName,'AssignState')">
                            <xsl:call-template
name="outputState"/>
                        </xsl:when>
                        <xsl:when
test="contains($PrimitiveBehaviorIDName,'BitwiseAnd')">
                            <xsl:call-template
name="outputBitwiseAnd"/>
                        </xsl:when>
                        <xsl:when
test="contains($PrimitiveBehaviorIDName,'AssignValue')">
```

```
                                    <xsl:call-template
name="outputAssignValue"/>
                              </xsl:when>
                              <xsl:otherwise>
                                    <xsl:value-of
select="@rdf:ID"/>(<xsl:apply-templates select="var:hasBehaviorInput |
var:hasBehaviorOutput"/><xsl:text>)</xsl:text>
                              </xsl:otherwise>
                        </xsl:choose>
                  </xsl:otherwise>
                  </xsl:choose>
      </xsl:template>

<!-- Used for printing out the arguments of a function. -->

      <xsl:template match="var:hasBehaviorInput">
            <xsl:variable name="InputIDName" select="substring-
after(@rdf:resource,'#')"/>
            <xsl:apply-templates select="//*[@rdf:ID=$InputIDName]"
mode="DisplayName"/>
            <xsl:call-template name="printComma"/>
      </xsl:template>

      <xsl:template match="var:hasBehaviorOutput">
            <xsl:variable name="OutputIDName" select="substring-
after(@rdf:resource,'#')"/>
            <xsl:apply-templates select="//*[@rdf:ID=$OutputIDName]"
mode="DisplayName"/>
            <xsl:call-template name="printComma"/>
      </xsl:template>

      <xsl:template name="printComma">
            <xsl:if test="position() &lt; last()">,</xsl:if>
      </xsl:template>

      <xsl:template match="var:Input" mode="DisplayName">
            <xsl:choose>
                  <xsl:when test="var:usesValueFrom">
                        <xsl:variable name="InputIDName"
select="substring-after(var:usesValueFrom/@rdf:resource,'#')"/>
                        <xsl:apply-templates
select="//*[@rdf:ID=$InputIDName]" mode="DisplayName"/>
                  </xsl:when>
                  <xsl:when test="var:DisplayName">
                        <xsl:value-of select="var:DisplayName"/>
                  </xsl:when>
                  <xsl:otherwise>

      <xsl:text>(NoDisplayValueFoundFor:</xsl:text><xsl:value-of
select="@rdf:ID"/><xsl:text>)</xsl:text>
                  </xsl:otherwise>
            </xsl:choose>
      </xsl:template>

      <xsl:template match="var:Output" mode="DisplayName">
            <xsl:choose>
                  <xsl:when test="var:usesValueFrom">
```

```
                        <xsl:variable name="IDName" select="substring-
after(var:usesValueFrom/@rdf:resource,'#')"/>
                        <xsl:apply-templates
select="//*[@rdf:ID=$IDName]" mode="DisplayName"/>
                  </xsl:when>
                  <xsl:when test="var:DisplayName">
                        <xsl:value-of select="var:DisplayName"/>
                  </xsl:when>
                  <xsl:when test="var:usedForValueOf">
                        <xsl:variable name="IDName" select="substring-
after(var:usedForValueOf/@rdf:resource,'#')"/>
                        <xsl:apply-templates
select="//*[@rdf:ID=$IDName]" mode="DisplayName"/>
                  </xsl:when>
                  <xsl:otherwise>

      <xsl:text>(NoDisplayValueFoundFor:</xsl:text><xsl:value-of
select="@rdf:ID"/><xsl:text>)</xsl:text>
                  </xsl:otherwise>
            </xsl:choose>
      </xsl:template>

<!-- Applies template for Composite Behavior and its next sibling
component, if any.-->

      <xsl:template match="beh:CompositeBehavior">
                  <xsl:variable name="CompositeBehaviorIDName"
select="@rdf:ID"/>
                  <xsl:choose>
                        <xsl:when
test="contains($CompositeBehaviorIDName,'JSAF_vmove_')">
                              <xsl:value-of select="substring-
before(substring-after(@rdf:ID,'JSAF_'),'_CB-')"/>(<xsl:apply-templates
select="var:hasBehaviorInput | var:hasBehaviorOutput"/><xsl:text>);
                              </xsl:text>
                        </xsl:when>
                        <xsl:otherwise>
                              <xsl:value-of
select="@rdf:ID"/>(<xsl:apply-templates select="var:hasBehaviorInput |
var:hasBehaviorOutput"/><xsl:text>);
                              </xsl:text>
                        </xsl:otherwise>
                  </xsl:choose>

                  <xsl:variable name="componentIDNext"
select="substring-after(beh:hasNextComponent/@rdf:resource,'#')"/>
                  <xsl:if test="$componentIDNext">
                        <xsl:apply-templates
select="//*[@rdf:ID=$componentIDNext]"/>
                  </xsl:if>
      </xsl:template>

<!-- Applies template for Composite Behavior and its next sibling
component, if any.-->

      <xsl:template match="beh:CompositeBehavior" mode="Conditional">
```

```
                  <xsl:variable name="CompositeBehaviorIDName"
select="@rdf:ID"/>
                  <xsl:choose>
                        <xsl:when
test="contains($CompositeBehaviorIDName,'JSAF_vmove_')">
                              <xsl:value-of select="substring-
before(substring-after(@rdf:ID,'JSAF_'),'_CB-')"/>(<xsl:apply-templates
select="var:hasBehaviorInput |
var:hasBehaviorOutput"/><xsl:text>)</xsl:text>
                        </xsl:when>
                        <xsl:otherwise>
                              <xsl:value-of
select="@rdf:ID"/>(<xsl:apply-templates select="var:hasBehaviorInput |
var:hasBehaviorOutput"/><xsl:text>)</xsl:text>
                        </xsl:otherwise>
                  </xsl:choose>
      </xsl:template>

<!-- Applies template for Conditional Branch and its Conditional
Expression, TrueBranchPath and FalseBranchPath, if any -->

      <xsl:template match="beh:ConditionalBranch">
                  <xsl:call-template name="outputIf"/>
                  <xsl:variable name="conditionalExpressionID"
select="substring-
after(beh:hasConditionalExpression/@rdf:resource,'#')"/>
                  <xsl:apply-templates
select="//*[@rdf:ID=$conditionalExpressionID]"/><xsl:text>)</xsl:text>
                  <xsl:variable name="trueBranchPathID"
select="substring-after(beh:hasTrueBranchPath/@rdf:resource,'#')"/>
                  <xsl:if test="$trueBranchPathID">
                        {
                        <xsl:apply-templates
select="//*[@rdf:ID=$trueBranchPathID]"/>
                  </xsl:if>
                        }
                  <xsl:variable name="falseBranchPathID"
select="substring-after(beh:hasFalseBranchPath/@rdf:resource,'#')"/>
                  <xsl:if test="$falseBranchPathID"> else {
                        <xsl:apply-templates
select="//*[@rdf:ID=$falseBranchPathID]"/>
                        }
                  </xsl:if>
      </xsl:template>

<!-- Applies template for type of Conditional Expression -->

      <xsl:template match="beh:ConditionalExpression">
                  <xsl:variable name="ConditionalPredicateID"
select="substring-
after(beh:hasConditionalPredicate/@rdf:resource,'#')"/>
                  <xsl:variable name="InputID" select="substring-
after(var:hasInput/@rdf:resource,'#')"/>
                  <xsl:variable name="ConditionalExpressionStatementID"
select="beh:ConditionalExpressionStatement"/>
                  <xsl:variable name="DefaultValueID"
select="beh:DefaultValue"/>
```

```
                  <xsl:choose>
                        <xsl:when test="$ConditionalPredicateID">
                              <xsl:apply-templates
select="//*[@rdf:ID=$ConditionalPredicateID]" mode="Conditional"/>
                        </xsl:when>
                        <xsl:when test="$InputID">
                              <xsl:apply-templates
select="//*[@rdf:ID=$InputID]" mode="DisplayName"/>
                        </xsl:when>
                        <xsl:when
test="$ConditionalExpressionStatementID">
                              <xsl:value-of
select="$ConditionalExpressionStatementID"/>
                        </xsl:when>
                        <xsl:when test="$DefaultValueID">
                              <xsl:value-of select="$DefaultValueID"/>
                        </xsl:when>
                  </xsl:choose>
      </xsl:template>

<!-- Applies template for type of UnarySentence -->

      <xsl:template match="beh:UnarySentence">
                  <xsl:variable name="UnaryOperatorTypeID"
select="beh:hasUnaryOperatorType/@rdf:resource"/>
                  <xsl:variable name="UnaryExpressionID"
select="substring-after(beh:hasUnaryExpression/@rdf:resource,'#')"/>
                  <xsl:if test="$UnaryExpressionID">
                        <xsl:choose>
                              <xsl:when
test="contains($UnaryOperatorTypeID,'Not')">
                                    <xsl:text>!(</xsl:text><xsl:apply-
templates
select="//*[@rdf:ID=$UnaryExpressionID]"/><xsl:text>)</xsl:text>
                              </xsl:when>
                              <xsl:when
test="contains($UnaryOperatorTypeID,'Inverse')">

      <xsl:text>(1/(</xsl:text><xsl:apply-templates
select="//*[@rdf:ID=$UnaryExpressionID]"/><xsl:text>))</xsl:text>
                              </xsl:when>
                              <xsl:otherwise>
                                    <xsl:text>Invalid UnarySentence
UnaryOperatorType</xsl:text>
                              </xsl:otherwise>
                        </xsl:choose>
                  </xsl:if>
      </xsl:template>

<!-- Applies template for type of BinarySentence -->

      <xsl:template match="beh:BinarySentence">
                  <xsl:variable name="BinaryOperatorTypeID"
select="substring-after(beh:hasBinaryOperatorType/@rdf:resource,'#')"/>
                  <xsl:variable name="LeftHandSideExpressionID"
select="substring-
after(beh:hasLeftHandSideExpression/@rdf:resource,'#')"/>
```

```
                <xsl:variable name="RightHandSideExpressionID"
select="substring-
after(beh:hasRightHandSideExpression/@rdf:resource,'#')"/>
                <xsl:if test="BinaryExpressionID">
                      <xsl:choose>
                            <xsl:when
test="contains($BinaryOperatorTypeID,'BinaryAnd')">
                                  <xsl:text>(</xsl:text><xsl:apply-
templates select="//*[@rdf:ID=$LeftHandSideExpressionID]"/><xsl:text>
&amp;&amp; </xsl:text><xsl:apply-templates
select="//*[@rdf:ID=$RightHandSideExpressionID]"/><xsl:text>)</xsl:text
>
                            </xsl:when>
                            <xsl:when
test="contains($BinaryOperatorTypeID,'BinaryOr')">
                                  <xsl:text>(</xsl:text><xsl:apply-
templates select="//*[@rdf:ID=$LeftHandSideExpressionID]"/><xsl:text>
|| </xsl:text><xsl:apply-templates
select="//*[@rdf:ID=$RightHandSideExpressionID]"/><xsl:text>)</xsl:text
>
                            </xsl:when>
                            <xsl:when
test="contains($BinaryOperatorTypeID,'LessThan')">
                                  <xsl:text>(</xsl:text><xsl:apply-
templates select="//*[@rdf:ID=$LeftHandSideExpressionID]"/><xsl:text>
&lt; </xsl:text><xsl:apply-templates
select="//*[@rdf:ID=$RightHandSideExpressionID]"/><xsl:text>)</xsl:text
>
                            </xsl:when>
                            <xsl:when
test="contains($BinaryOperatorTypeID,'LessThanOrEqual')">
                                  <xsl:text>(</xsl:text><xsl:apply-
templates select="//*[@rdf:ID=$LeftHandSideExpressionID]"/><xsl:text>
&lt;= </xsl:text><xsl:apply-templates
select="//*[@rdf:ID=$RightHandSideExpressionID]"/><xsl:text>)</xsl:text
>
                            </xsl:when>
                            <xsl:when
test="contains($BinaryOperatorTypeID,'Equal')">
                                  <xsl:text>(</xsl:text><xsl:apply-
templates select="//*[@rdf:ID=$LeftHandSideExpressionID]"/><xsl:text>
== </xsl:text><xsl:apply-templates
select="//*[@rdf:ID=$RightHandSideExpressionID]"/><xsl:text>)</xsl:text
>
                            </xsl:when>
                            <xsl:when
test="contains($BinaryOperatorTypeID,'NotEqual')">
                                  <xsl:text>(</xsl:text><xsl:apply-
templates select="//*[@rdf:ID=$LeftHandSideExpressionID]"/><xsl:text>
!= </xsl:text><xsl:apply-templates
select="//*[@rdf:ID=$RightHandSideExpressionID]"/><xsl:text>)</xsl:text
>
                            </xsl:when>
                            <xsl:when
test="contains($BinaryOperatorTypeID,'GreaterThanOrEqual')">
                                  <xsl:text>(</xsl:text><xsl:apply-
templates select="//*[@rdf:ID=$LeftHandSideExpressionID]"/><xsl:text>
```

```
>= </xsl:text><xsl:apply-templates
select="//*[@rdf:ID=$RightHandSideExpressionID]"/><xsl:text>)</xsl:text
>
                                    </xsl:when>
                                    <xsl:when
test="contains($BinaryOperatorTypeID,'GreaterThan')">
                                          <xsl:text>(</xsl:text><xsl:apply-
templates select="//*[@rdf:ID=$LeftHandSideExpressionID]"/><xsl:text> >
</xsl:text><xsl:apply-templates
select="//*[@rdf:ID=$RightHandSideExpressionID]"/><xsl:text>)</xsl:text
>
                                    </xsl:when>
                                    <xsl:otherwise>
                                          <xsl:text>Invalid BinarySentence
BinaryOperatorType</xsl:text>
                                    </xsl:otherwise>
                              </xsl:choose>
                        </xsl:if>
      </xsl:template>

<!-- Miscellanieous called and matched templates -->

<xsl:template match="var:Input">
      <xsl:call-template name="InputDisplayName"/>
</xsl:template>

<xsl:template name="outputSTART">`
START
`</xsl:template>

<xsl:template name="outputEND">`
END
`</xsl:template>

<xsl:template name="outputTick">`
<xsl:value-of select="substring-before($globalName,'_tick')"/>
`     tick</xsl:template>

<xsl:template name="outputParams">`
<xsl:value-of select="substring-before($globalName,'_params')"/>
`     params</xsl:template>

<xsl:template name="outputIf">      if (</xsl:template>

<xsl:template name="outputState">
            <xsl:variable name="assignedStateID" select="substring-
after(var:hasBehaviorInput/@rdf:resource,'#')"/>      ^<xsl:value-of
select="//*[@rdf:ID=$assignedStateID]/var:DisplayName"/>; <xsl:value-of
select="beh:Note"/>

</xsl:template>

<!-- This assumes that the two inputs are variables that are external
inputs to the Composite Behavior -->
<xsl:template name="outputBitwiseAnd">
            <xsl:variable name="BitwiseAndID1" select="substring-
after(var:hasBehaviorInput[1]/@rdf:resource,'#')"/>
```

```
            <xsl:variable name="BitwiseAndID2" select="substring-
after(var:hasBehaviorInput[2]/@rdf:resource,'#')"/>
            <xsl:apply-templates select="//*[@rdf:ID=$BitwiseAndID1]"
mode="BitwiseAnd"/> &amp;amp; <xsl:apply-templates
select="//*[@rdf:ID=$BitwiseAndID2]" mode="BitwiseAnd"/>

</xsl:template>

<xsl:template match="var:Input" mode="BitwiseAnd">
            <xsl:variable name="BitwiseAndIDInput" select="substring-
after(var:usesValueFrom/@rdf:resource,'#')"/>
            <xsl:value-of
select="//*[@rdf:ID=$BitwiseAndIDInput]/var:DisplayName"/>
</xsl:template>

<!-- This assumes that the output and input are a variable and an input
value that are external output and input to the Composite Behavior -->
<xsl:template name="outputAssignValue">
            <xsl:variable name="OutputIDName" select="substring-
after(var:hasBehaviorOutput/@rdf:resource,'#')"/>
            <xsl:variable name="InputIDName" select="substring-
after(var:hasBehaviorInput/@rdf:resource,'#')"/>
            <xsl:apply-templates select="//*[@rdf:ID=$OutputIDName]"
mode="DisplayName"/> = <xsl:apply-templates
select="//*[@rdf:ID=$InputIDName]" mode="DisplayName"/>
</xsl:template>

<!-- This assumes that the output is a variable and the input is a
function, both of which are external output and input to the Composite
Behavior -->
<xsl:template name="outputAssignFunctionValue">
            <xsl:variable name="OutputIDName" select="substring-
after(var:hasBehaviorOutput/@rdf:resource,'#')"/>
            <xsl:variable name="InputIDName" select="substring-
after(var:hasBehaviorInput/@rdf:resource,'#')"/>
            <xsl:apply-templates select="//*[@rdf:ID=$OutputIDName]"
mode="DisplayName"/> = <xsl:apply-templates
select="//*[@rdf:ID=$InputIDName]" mode="DisplayName"/>
</xsl:template>

<xsl:template name="InputDisplayName">
      <xsl:value-of select="string(var:DisplayName)"/></xsl:template>

</xsl:stylesheet>
```

Attachment 24–XSLT Transformed
JSAF_vmove_DRIVING_params_CB.rdf

```
`
DRIVING
`     params
            {
                        if (parameters->termination &
VMOVE_TERM_IMMEDIATE)
                        {
                              ^END; we were told to die immediately
                        }
                        if (vmove_no_input_route(priv,parameters))
                        {
                              ^CHASING; make our own route
                        }

      vmove_update_private(vehicle_id,priv,state,parameters,task_entry)
;
            }
```

Attachment 25–XSLT Transformed JSAF_vmove_START_CB.rdf

```
START
`
        {
                priv->chase = False;

                        if (!(state->cell))
                        {
                        state->cell = wld_get_playbox_center_cell( );

                        }

vmove_update_private(vehicle_id,priv,state,parameters,task_entry);
                        ^DISABLED; always disabled when in start
        }
```